

**SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY**

**(A Constituent college of Sri Siddhartha**

**Academyof HigherEducation)**

**Department of Computer Science & Engineering**

**Report on “DBMS Lab Programs”& Mini Project on “Hotel Management System”**

***Submitted by***

**ACHYUTH.K.P(17CS002)**

**APURV KUMAR(17CS008)**

**CHANDANA.K.G(17CS018)**

**CHANDNA SINGH(17CS019)**

**UNDER THE GUIDANCE OF:**

**T.N Sindhu M.Tech Dr.S.Kumaraswamy Ph.D**

Assistant Professor Professor

Dept.of C.S.E. Dept. of C.S.E.

S.S.I.T,Tumkur S.S.I.T,Tumkur



***Certificate***

This is to certify that the course **DBMS Lab Programs with miniproject “Hotel Management System”(CS5L02)**has been completed successfully in fifth semester of Bachelor of Engineering in Computer Science and Engineering of Sri Siddhartha Academy of Higher Education during the academic year 2019-20.

**UNDER THE GUIDANCE OF:**

**T.N.Sindhu M.Tech Dr.S.Kumaraswamy Ph.D**

Assistant Professor Professor

Dept. of C.S.E . Dept. of C.S.E.

S.S.I.T,Tumkur S.S.I.T,Tumkur

**Dr.M.Siddappa Ph.D**

Professor & Head,

Dept.of C.S.E.

S.S.I.T.,Tumkur

Examiner 1 Name: Examiner 2 Name:

Signature: Signature:

**PART -A**

**TABLE CONTENT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **sr.** |  | **title** | **Page** |  |
| **no** |  |  | **no** |  |
|  |  |  |  |  |
| **1** |  | **INTRODUCTION TO SQL** | **1** |  |
|  |  |  |  |  |
| **1.1** |  | **DATA DEFINATION,CONSTRAINTS,AND** |  |  |
|  |  | **SCHEMAS** |  |  |
|  |  |  |  |  |
| **1.2** |  | **CREATE ,ALTER,AND DROP** | **2-3** |  |
| **1.3** |  | **Basic queries in SQL** | **4-16** |  |
| **1.4** |  | **Simple SQL queries** |  |
|  |  |  |
| **1.5** |  | **Aliases,\*,distinct,empty,where-clause** |  |  |
|  |  |  |  |  |
| **1.6** |  | **Unspecified WHERE –clause** |  |  |
|  |  |  |  |  |
| **1.7** |  | **SET Operations,Grouping** |  |  |
|  |  |  |  |  |
| **1.8** |  | **Correlated Nested Queries** |  |  |
|  |  |  |  |  |
| **1.9** |  | **VIEWS in SQL** |  |  |
|  |  |  |  |  |
| **2** |  | **SCHEMA FOR LIBRARY DATABASE** | **16-** |  |
| **2.1** |  | **Entity-relationship diagram,schema diagram** | **29** |  |
| **2.2** |  | **Table creation,Table insertion** |  |
|  |  |  |
| **2.3** |  | **Queries** |  |  |
|  |  |  |  |  |
| **3** |  | **SCHEMA FOR ORDER DATABASE** | **30-** |  |
| **3.1** |  | **Entity-relationship diagram,schema diagram** |  |
|  | **37** |  |
| **3.2** |  | **Table creation,Table insertion** |  |
|  |  |  |
|  |  |  |  |  |
| **3.3** |  | **Queries** |  |  |
| **4** |  | **SCHEMA FOR MOVIE DATABASE** | **38-** |  |
| **4.1** |  | **Entity-relationship diagram,schema diagram** |  |
|  | **46** |  |
|  |  |  |  |
| **4.2** |  | **Table creation,Table insertion** |  |
|  |  |  |
| **4.3** |  | **Queries** |  |  |
|  |  |  |  |  |
| **5** |  | **SCHEMA FOR COMPANY DATABASE** | **47-** |  |
| **5.1** |  | **Entity-relationship diagram,schema diagram** | **60** |  |
| **5.2** |  | **Table creation,Table insertion** |  |  |
| **5.3** |  | **Queries** |  |  |
|  |  | |  |  |

**1.INTRODUCTION TO SQL**

Pronounced as SEQUEL: Structured English QUERY Language

|  |  |  |
| --- | --- | --- |
|  | Pure non-procedural query language |  |
| Designed and developed by IBM, Implemented by Oracle |  |
|  |  |
| 1978 System/R IBM- 1st Relational DBMS |  |
|  |  |
| 1979 Oracle and Ingres |  |
|  |  |
| 1982 SQL/DS and DB2 IBM |  |
|  |  |
| Accepted by both ANSI + ISO as **Standard Query Language** for any RDBMS |  |
|  |  |
| SQL86 (SQL1) : first by ANSI and ratified by ISO (SQL-87), minor revision on 89 |  |
|  |  |
|  | (SQL-89) |  |
| SQL92 (SQL2) : major revision |  |
|  |  |
| SQL99 (SQL3) : add recursive query, trigger, some OO features, and non-scholar type |  |
|  |  |
| SQL2003 : XML, Window functions, and sequences (Not free) |  |
|  |  |
| Supports all the three sublanguages of DBMS: **DDL, DML, DCL** |  |
|  |  |
| Supports Aggregate functions, String Manipulation functions, Set theory operations, |  |
|  |  |
|  | Date Manipulation functions, rich set of operators ( IN, BETWEEN, LIKE, IS NULL, |  |
| EXISTS) |  |
|  |  |
| Supports REPORT writing features and Forms for designing GUI based applications |  |
|  |  |

**1.1 DATA DEFINITION, CONSTRAINTS, AND SCHEMA CHANGES**

Used to CREATE, ALTER, and DROP the descriptions of the database tables (relations)

**Data Definition in SQL**

**1.2 CREATE, ALTER and DROP**

table…………………………………….……relation

row……………………………………..…….tuple

column………………………………….……attribute

**DATA TYPES**

|  |  |  |
| --- | --- | --- |
|  | Numeric: NUMBER, NUMBER(s,p), INTEGER, INT, FLOAT, DECIMAL |  |
| Character: CHAR(n), VARCHAR(n), VARCHAR2(n), CHAR VARYING(n) |  |
|  |  |
| Bit String: BLOB, CLOB |  |
|  |  |
| Boolean: true, false, and null |  |
|  |  |



SSIT, Tumakuru Page 1

*DBMS Lab with mini project* *Dept. of CSE*

|  |  |  |
| --- | --- | --- |
|  | Date and Time: DATE (YYYY-MM-DD) TIME( HH:MM:SS) |  |
| Timestamp: DATE + TIME |  |
|  |  |
| USER Defined types |  |
|  |  |

**CREATE SCHEMA**

Specifies a new database schema by giving it a name

Ex: CREATE SCHEMA COMPANY AUTHORIZATION Jsmith;

**CREATE TABLE**

Specifies a new base relation by giving it a name, and specifying each of its attributes and

* their data types
* Syntax of CREATE Command:

**CREATE TABLE <***table name>*( <Attribute*A*1> <Data Type*D*1> [<

Constarints>], <Attribute *A*2> <Data Type *D*2> [< Constarints>],



 *…….*

<Attribute *A*n> <Data Type *D*n> [< Constarints>], [<integrity-constraint1>, <integrity-constraint k> ] ); - A constraint NOT NULL may be specified on an attribute A constraint NOT NULL may be specified on an attribute Ex: CREATE TABLE DEPARTMENT ( DNAME VARCHAR(10) NOT NULL, DNUMBER INTEGER NOT NULL,

MGRSSN CHAR(9), MGRSTARTDATE CHAR(9) );

Specifying the unique, primary key attributes, secondary keys, and referential integrity

constraints (foreign keys).



Ex: CREATE TABLE DEPT ( DNAME





VARCHAR(10) NOT NULL,

DNUMBER INTEGER NOT NULL,

MGRSSN CHAR(9),

MGRSTARTDATE CHAR(9),

PRIMARY KEY (DNUMBER),

UNIQUE (DNAME),

FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN));



SSIT, Tumakuru Page 2

*DBMS Lab with mini project* *Dept. of CSE*

We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential

|  |  |  |
| --- | --- | --- |
|  | integrity constraints (foreign keys) |  |
| Ex: CREATE TABLE DEPT ( DNAME |  |
|  |  |
|  | VARCHAR(10) NOT NULL, |  |
|  | DNUMBER INTEGER NOT NULL, |  |
| MGRSSN CHAR(9), MGRSTARTDATE CHAR(9), |  |
|  |  |
| PRIMARY KEY (DNUMBER), |  |
|  |  |
|  | UNIQUE (DNAME), |  |
|  | FOREIGN KEY (MGRSSN) REFERENCES EMP |  |
|  | ON DELETE SET DEFAULT ON UPDATE CASCADE); |  |

**DROP TABLE**

Used to remove a relation (base table) and its definition.



The relation can no longer be used in queries, updates, or any other commands since its description no longer exists

**Example:** DROP TABLE DEPENDENT;

**ALTER TABLE:**

Used to add an attribute to/from one of the base relations drop constraint -- The new attribute will have NULLs in all the tuples of the relation right after the command is

executed; hence, the NOT NULL constraint is *not allowed* for such an attribute.



**Example:** ALTER TABLE EMPLOYEE ADD JOB VARCHAR2 (12);



The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

**DROP A COLUMN (AN ATTRIBUTE)**

ALTER TABLE COMPANY.EMPLOYEE DROP ADDRESS CASCADE; All constraints and views that reference the column are dropped automatically, along with the column. ALTER TABLE COMPANY.EMPLOYEE DROP ADDRESS RESTRICT; Successful if no views or constraints reference the column. ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN DROP DEFAULT;



 ALTER TABLE COMPANY.DEPARTMENT ALTER MGRSSN SET DEFAULT “333445555”;



SSIT, Tumakuru Page 3

*DBMS Lab with mini project* *Dept. of CSE*

**1.3BASIC QUERIES IN SQL**

|  |  |  |
| --- | --- | --- |
|  | SQL has one basic statement for retrieving information from a database; the SLELECT |  |
|  | statement |  |
| This is *not the same as* the SELECT operation of the relational algebra |  |
|  |  |
| Important distinction between SQL and the formal relational model; |  |
|  |  |
|  |  |

SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values



Hence, an SQL relation (table) is a *multi-set* (sometimes called a bag) of tuples; it is *not* a

set of tuples



SQL relations can be constrained to be sets by using the CREATE UNIQUE INDEX

command, or by using the DISTINCT option



|  |  |  |
| --- | --- | --- |
|  | Basic form of the SQL SELECT statement is called a *mapping* of a *SELECT-FROM-* |  |
|  | *WHERE block* |  |
| SELECT <attribute list> FROM <table list> WHERE <condition> |  |
|  |  |
| <attribute list> is a list of attribute names whose values are to be retrieved by the query |  |
|  |  |
| <table list > is a list of the relation names required to process the query |  |
|  |  |
| <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved |  |
|  |  |
|  | by the query |  |

**1.4 SIMPLE SQL QUERIES**

Basic SQL queries correspond to using the following operations of the relational algebra:

SELECT

PROJECT

JOIN

All subsequent examples uses COMPANY database as shown below:

**Example of a simple query on one relation**

**Query 1: Retrieve the birth date and address of the employee whose name is 'John B.**

**Smith'.**

Q1: SELECT BDATE, ADDRESS FROM EMPLOYEE

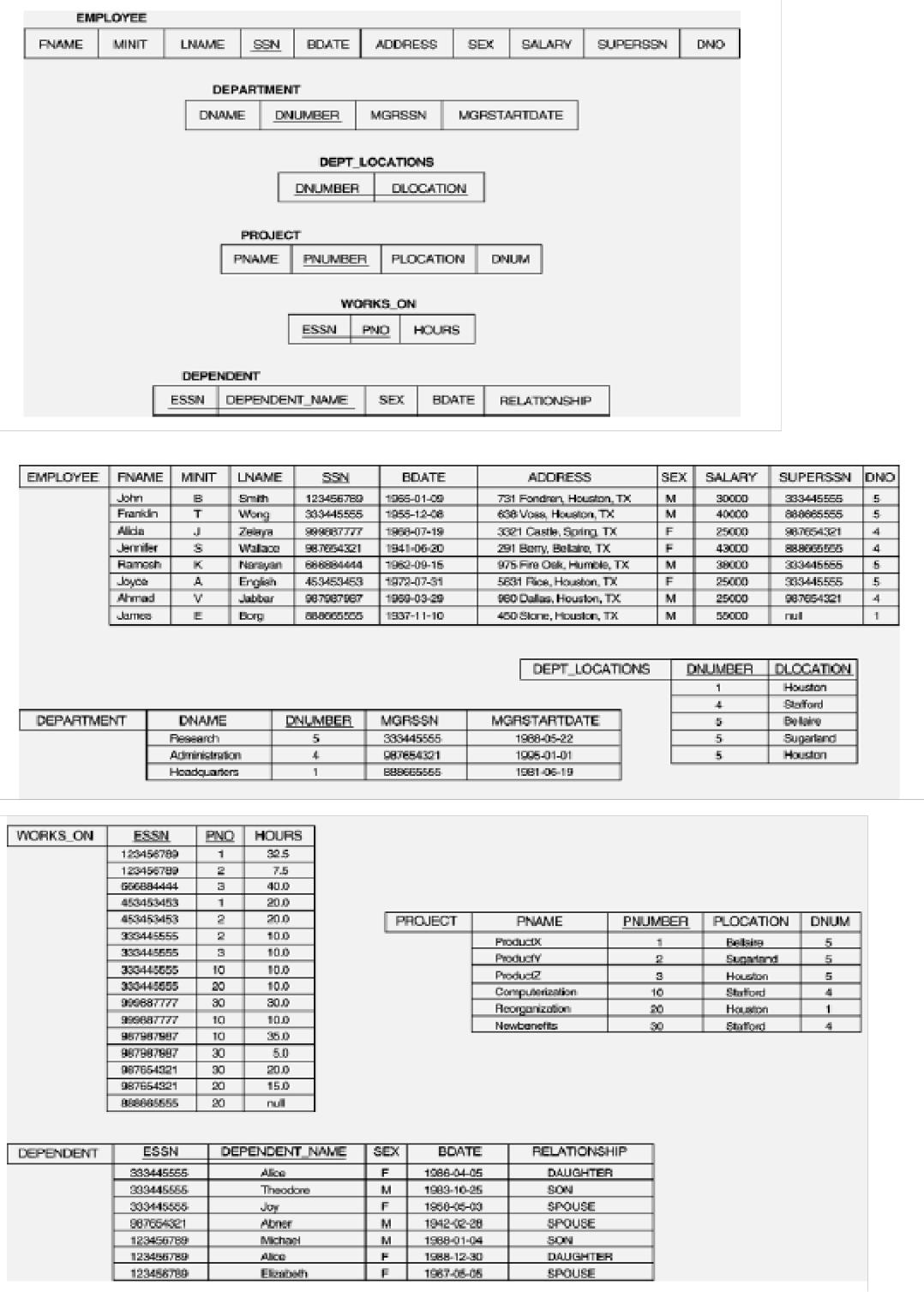
WHERE FNAME='John' AND MINIT='B’ AND LNAME='Smith’ ;

Similar to a SELECT-PROJECT pair of relational algebra operations: The SELECT-clause specifies the projection attributes and the WHERE-clause specifies the selection condition However, the result of the query may contain duplicate tuples



SSIT, Tumakuru Page 4

*DBMS Lab with mini project* *Dept. of CSE*



**Example of a simple query on two relations**

**Query 2: Retrieve the name and address of all employees who work for the 'Research'**

**department.**



SSIT, Tumakuru Page 5

*DBMS Lab with mini project* *Dept. of CSE*

Q2: SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE, DEPARTMENT WHERE DNAME='Research' AND DNUMBER=DNO ;

Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations (DNAME='Research') is a selection condition (corresponds to a SELECT operation in relational algebra) (DNUMBER=DNO) is a join condition (corresponds to a JOIN operation in relational algebra)

**Example of a simple query on three relations**

**Query 3: For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.**

Q3: SELECT PNUMBER, DNUM, LNAME, BDATE, ADDRESS FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE DNUM=DNUMBER AND MGRSSN=SSN AND PLOCATION='Stafford' ;

In Q3, there are two join conditions The join condition DNUM=DNUMBER relates a project to its controlling department The join condition MGRSSN=SSN relates the controlling department to the employee who manages that department

**1.5 ALIASES, \* AND DISTINCT, EMPTY WHERE-CLAUSE**

In SQL, we can use the same name for two (or more) attributes as long as the attributes

are in different relations



A query that refers to two or more attributes with the same name must qualify the attribute name with the relation name by prefixing the relation name to the attribute name

**Example:** EMPLOYEE.LNAME, DEPARTMENT.DNAME



Some queries need to refer to the same relation twice. In this case, aliases are given to the relation name

**Example**

**Query 4: For each employee, retrieve the employee's name, and the name of his or her**

**immediate supervisor.**

Q4: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME FROM EMPLOYEE E S WHERE E.SUPERSSN=S.SSN ;

In Q4, the alternate relation names E and S are called aliases or tuple variables for the EMPLOYEE relation We can think of E and S as two different copies of EMPLOYEE; E represents employees in role of supervisees and S represents employees in role of supervisors



SSIT, Tumakuru Page 6

*DBMS Lab with mini project* *Dept. of CSE*

Aliasing can also be used in any SQL query for convenience. Can also use the AS keyword to specify aliases

Q4: SELECT E.FNAME, E.LNAME, S.FNAME, S.LNAME FROM EMPLOYEE AS E, EMPLOYEE AS S WHERE E.SUPERSSN=S.SSN ;

**1.6 UNSPECIFIED WHERE-clause**

A missing WHERE-clause indicates no condition; hence, all tuples of the relations in the

FROM-clause are selected. This is equivalent to the condition WHERE TRUE Example:

**Query 5: Retrieve the SSN values for all employees.**

Q4: SELECT SSN FROM EMPLOYEE;

If more than one relation is specified in the FROM-clause and there is no join condition, then the

CARTESIAN PRODUCT of tuples is selected

Example:

Q6: SELECT SSN, DNAME FROM EMPLOYEE, DEPARTMENT;

**Note:** It is extremely important not to overlook specifying any selection and join conditions inthe WHERE-clause; otherwise, incorrect and very large relations may result

**USE OF \***

To retrieve all the attribute values of the selected tuples, a \* is used, which stands for all the

attributes

Examples:

R**etrieve all the attribute values of EMPLOYEES who work in department 5.**

Q1a: SELECT \* FROM EMPLOYEE WHERE DNO=5;

**Retrieve all the attributes of an employee and attributes of DEPARTMENT he works in for every employee of ‘Research’ department.**

Q1b: SELECT \* FROM EMPLOYEE, DEPARTMENT WHERE DNAME='Research' AND DNO=DNUMBER ;

**USE OF DISTINCT**

SQL does not treat a relation as a set; duplicate tuples can appear. To eliminate duplicate tuples in a query result, the keyword DISTINCT is used



SSIT, Tumakuru Page 7

*DBMS Lab with mini project* *Dept. of CSE*

Example: the result of **Q1c** may have duplicate SALARY values whereas **Q1d** does not have any duplicate values

Q1c: SELECT SALARY FROM EMPLOYEE Q1d: SELECT **DISTINCT** SALARY FROM EMPLOYEE ;

**1.7 SET OPERATIONS**

SQL has directly incorporated some set operations such as union operation (UNION), set difference (MINUS) and intersection (INTERSECT) operations. The resulting relations of these set operations are sets of tuples; duplicate tuples are eliminated from the result. The set operations apply only to union compatible relations; the two relations must have the same attributes and the attributes must appear in the same order

**Query 6: Make a list of all project numbers for projects that involve an employee whose last name is 'Smith' as a worker or as a manager of the department that controls the project.**

Q6: (SELECT PNAME FROM PROJECT, DEPARTMENT, EMPLOYEE WHERE DNUM=DNUMBER AND MGRSSN=SSN AND LNAME='Smith')

**UNION**

(SELECT PNAME FROM PROJECT, WORKS\_ON, EMPLOYEE WHERE PNUMBER=PNO AND ESSN=SSN AND NAME='Smith') ;

**NESTING OF QUERIES**

A complete SELECT query, called a nested query, can be specified within the WHERE-clause of another query, called the outer query. Many of the previous queries can be specified in an alternative form using nesting

**Query 7: Retrieve the name and address of all employees who work for the 'Research' department.**

Q7: SELECT FNAME, LNAME, ADDRESS FROM EMPLOYEE WHERE DNO **IN** (SELECT DNUMBER FROM DEPARTMENT WHERE DNAME='Research' ) ;

**Note:** The nested query selects the number of the 'Research' department. The outer query selectsan EMPLOYEE tuple if its DNO value is in the result of either nested query. The comparison operator IN compares a value v with a set (or multi-set) of values V, and evaluates to TRUE if v is one of the elements in V



SSIT, Tumakuru Page 8

*DBMS Lab with mini project* *Dept. of CSE*

In general, we can have several levels of nested queries. A reference to an unqualified attribute refers to the relation declared in the innermost nested query. In this example, the nested query is not correlated with the outer query

**1.8 CORRELATED NESTED QUERIES**

If a condition in the WHERE-clause of a nested query references an attribute of a relation declared in the outer query, the two queries are said to be correlated. The result of a correlated nested query is different for each tuple (or combination of tuples) of the relation(s) the outer query

**Query 8: Retrieve the name of each employee who has a dependent with the same first name as the employee**.

Q8: SELECT E.FNAME, E.LNAME FROM EMPLOYEE AS E WHERE E.SSN **IN** (SELECT ESSN FROM DEPENDENT WHERE ESSN=E.SSN AND E.FNAME=DEPENDENT\_NAME) ;

In Q8, the nested query has a different result in the outer query. A query written with nested SELECT... FROM… WHERE... blocks and using the **= or IN** comparison operators can *always* be expressed as a single block query. For example, Q7 may be written as in Q8a

Q8a: SELECT E.FNAME, E.LNAME FROM EMPLOYEE E, DEPENDENT D WHERE E.SSN=D.ESSN AND E.FNAME=D.DEPENDENT\_NAME ;

**THE EXISTS FUNCTION**

EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not. We can formulate Query 8 in an alternative form that uses EXISTS.

Q8b: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE **EXISTS** (SELECT \* FROM DEPENDENT WHERE SSN=ESSN AND FNAME=DEPENDENT\_NAME) ;

**Query 9: Retrieve the names of employees who have no dependents***.*

Q9: SELECT FNAME, LNAME FROM EMPLOYEE WHERE **NOT EXISTS**

(SELECT \* FROM DEPENDENT WHERE SSN=ESSN);

**Note:** In Q9, the correlated nested query retrieves all DEPENDENT tuples related to anEMPLOYEE tuple. If none exist, the EMPLOYEE tuple is selected

**EXPLICIT SETS**



SSIT, Tumakuru Page 9

*DBMS Lab with mini project* *Dept. of CSE*

It is also possible to use an explicit (enumerated) set of values in the WHERE-clause rather than a nested query

**Query 10: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.**

Q10: SELECT DISTINCT ESSN FROM WORKS\_ON WHERE PNO **IN (1, 2, 3) ;**

**NULLS IN SQL QUERIES**

SQL allows queries that check if a value is NULL (missing or undefined or not applicable). SQL uses IS or IS NOT to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate.

**Query 11: Retrieve the names of all employees who do not have supervisors.**

Q11: SELECT FNAME, LNAME FROM EMPLOYEE

WHERE SUPERSSN IS NULL **;**

**Note:** If a join condition is specified, tuples with NULL values for the join attributes are notincluded in the result

**AGGREGATE FUNCTIONS**

Include COUNT, SUM, MAX, MIN, and AVG

**Query 12: Find the maximum salary, the minimum salary, and the average salary among all employees.**

Q12: SELECT **MAX (SALARY), MIN(SALARY), AVG(SALARY)**

FROM EMPLOYEE;

**Note:** Some SQL implementations may not allow more than one function in the SELECT-clause

**Query 13: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.**

Q13: SELECT **MAX (SALARY), MIN(SALARY), AVG(SALARY)** FROM

EMPLOYEE, DEPARTMENT WHERE DNO=DNUMBER AND DNAME='Research' ;

**Queries 14 and 15: Retrieve the total number of employees in the company (Q14), and the number of employees in the 'Research' department (Q15).**

Q14: SELECT **COUNT (\*)** FROM EMPLOYEE ;

Q15: SELECT **COUNT (\*)** FROM EMPLOYEE, DEPARTMENT

WHERE DNO=DNUMBER AND DNAME='Research’;



SSIT, Tumakuru Page 10

*DBMS Lab with mini project* *Dept. of CSE*

**GROUPING**

In many cases, we want to apply the aggregate functions to subgroups of tuples in a

relation



Each subgroup of tuples consists of the set of tuples that have the same value for the

grouping attribute(s)



The function is applied to each subgroup independently



SQL has a GROUP BY-clause for specifying the grouping attributes, which must also appear in the SELECT-clause

**Query 16: For each department, retrieve the department number, the number of**

**employees in the department, and their average salary.**

Q16: SELECT DNO, COUNT (\*), AVG (SALARY)

FROM EMPLOYEE **GROUP BY** DNO ;

In Q16, the EMPLOYEE tuples are divided into groups. Each group having the same

value for the grouping attribute DNO



The COUNT and AVG functions are applied to each such group of tuples separately



The SELECT-clause includes only the grouping attribute and the functions to be applied

on each group of tuples



A join condition can be used in conjunction with grouping

**Query 17: For each project, retrieve the project number, project name, and the number of**

**employees who work on that project.**

Q17: SELECT PNUMBER, PNAME, COUNT (\*)

FROM PROJECT, WORKS\_ON

WHERE PNUMBER=PNO

GROUP BY PNUMBER, PNAME ;

**THE HAVING-CLAUSE**

Sometimes we want to retrieve the values of these functions for only those groups that satisfy certain conditions. The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

**Query 18: For each project on which more than two employees work, retrieve the project number, project name, and the number of employees who work on that project.**



SSIT, Tumakuru Page 11

*DBMS Lab with mini project* *Dept. of CSE*

Q18: SELECT PNUMBER, PNAME, COUNT

* FROM PROJECT, WORKS\_ON WHERE PNUMBER=PNO GROUP BY PNUMBER, PNAME

HAVING COUNT (\*) > 2 ;

**SUBSTRING COMPARISON**

The LIKE comparison operator is used to compare partial strings. Two reserved characters are used: **'%'** (or '\*' in some implementations) replaces an arbitrary number of characters, and **'\_'** replaces a single arbitrary character.

**Query 19: Retrieve all employees whose address is in Houston, Texas. Here, the value of the**

**ADDRESS attribute must contain the substring 'Houston,TX‘ in it.**

Q19: SELECT FNAME, LNAME

FROM EMPLOYEE WHERE ADDRESS LIKE '%Houston,TX%'

**Query 20: Retrieve all employees who were born during the 1950s.**

Here, '5' must be the 8th character of the string (according to our format for date), so the BDATE value is '\_\_\_\_\_\_\_5\_', with each underscore as a place holder for a single arbitrary character.

Q20: SELECT FNAME, LNAME

FROM EMPLOYEE WHERE BDATE **LIKE** '**\_\_\_\_\_\_\_**5**\_**’ ;

**Note:** The LIKE operator allows us to get around the fact that each value is considered atomicand indivisible. Hence, in SQL, character string attribute values are not atomic

**ARITHMETIC OPERATIONS**

The standard arithmetic operators '+', '-'. '\*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result

**Query 21: Show the effect of giving all employees who work on the 'ProductX' project a**

**10% raise.**

Q21: SELECT FNAME, LNAME, 1.1\*SALARY

FROM EMPLOYEE, WORKS\_ON, PROJECT



SSIT, Tumakuru Page 12

*DBMS Lab with mini project* *Dept. of CSE*

WHERE SSN=ESSN

AND PNO=PNUMBER AND PNAME='ProductX’ ;

**ORDER BY**

The ORDER BY clause is used to sort the tuples in a query result based on the values of some attribute(s)

**Query 22: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.**

Q22: SELECT DNAME, LNAME, FNAME, PNAME

FROM DEPARTMENT, EMPLOYEE, WORKS\_ON, PROJECT

WHERE DNUMBER=DNO

AND SSN=ESSN

AND PNO=PNUMBER

ORDER BY DNAME, LNAME ;

The default order is in ascending order of values. We can specify the keyword DESC if we want a descending order; the keyword ASC can be used to explicitly specify ascending order, even though it is the default

Ex: ORDER BY DNAME **DESC**, LNAME **ASC**, FNAME **ASC ;**

**MORE EXAMPLE QUERIES:**

**Query 23: Retrieve the names of all employees who have two or more dependents.**

Q23: SELECT LNAME, FNAME FROM

EMPLOYEE

WHERE (SELECT COUNT (\*) FROM DEPENDENT

WHERE SSN=ESSN) ≥ 2);

**Query 24: List the names of managers who have least one dependent.**

Q24: SELECT FNAME, LNAME



SSIT, Tumakuru Page 13

*DBMS Lab with mini project* *Dept. of CSE*

FROM EMPLOYEE

WHERE EXISTS (SELECT \* FROM DEPENDENT WHERE SSN=ESSN)

AND EXISTS ( SELECT \* FROM DEPARTMENT WHERE SSN=MGRSSN ); **SPECIFYING UPDATES IN SQL**

There are three SQL commands to modify the database: **INSERT**, **DELETE**, and **UPDATE.**

**INSERT**

In its simplest form, it is used to add one or more tuples to a relation



Attribute values should be listed in the same order as the attributes were specified in the **CREATE TABLE** command

**Example:**

INSERT INTO EMPLOYEE VALUES ('Richard','K','Marini', '653298653', '30-DEC-52', '98 Oak Forest,Katy,TX', 'M', 37000,'987654321', 4 ) ;

An alternate form of INSERT specifies explicitly the attribute names that correspond to the values in the new tuple. Attributes with NULL values can be left out

**Example:** Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME,and SSN attributes.

INSERT INTO EMPLOYEE (FNAME, LNAME, SSN)VALUES ('Richard', 'Marini', '653298653') ;

**Important Note**: Only the constraints specified in the DDL commands are automaticallyenforced by the DBMS when updates are applied to the database. Another variation of INSERT allows insertion of multiple tuples resulting from a **query** into a relation

**Example:** Suppose we want to create a temporary table that has the name, number of employees,and total salaries for each department. A table DEPTS\_INFO is created first, and is loaded with the summary information retrieved from the database by the query.

CREATE TABLE DEPTS\_INFO

(DEPT\_NAME VARCHAR (10),

NO\_OF\_EMPS INTEGER, TOTAL\_SAL INTEGER);

INSERT INTO DEPTS\_INFO (DEPT\_NAME, NO\_OF\_EMPS, TOTAL\_SAL) SELECT DNAME, COUNT (\*), SUM (SALARY) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO GROUP BY DNAME ;



SSIT, Tumakuru Page 14

*DBMS Lab with mini project* *Dept. of CSE*

**Note:** The DEPTS\_INFO table may not be up-to-date if we change the tuples in either theDEPARTMENT or the EMPLOYEE relations *after* issuing the above. We have to create a view (see later) to keep such a table up to date.

**DELETE**

Removes tuples from a relation. Includes a WHERE-clause to select the tuples to be

deleted



Referential integrity should be enforced



Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a

referential integrity constraint)



A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table



The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

Examples:

1. DELETE FROM EMPLOYEE WHERE LNAME='Brown’;
2. DELETE FROM EMPLOYEE WHERE SSN='123456789’;
3. DELETE FROM EMPLOYEE WHERE DNO IN (SELECT DNUMBER FROM DEPARTMENT WHERE DNAME='Research');
4. DELETE FROM EMPLOYEE;

**UPDATE**

|  |  |  |
| --- | --- | --- |
|  | Used to modify attribute values of one or more selected tuples |  |
| A WHERE-clause selects the tuples to be modified |  |
|  |  |
| An additional SET-clause specifies the attributes to be modified and their new values |  |
|  |  |
| Each command modifies tuples *in the same relation* |  |
|  |  |
| Referential integrity should be enforced |  |
|  |  |

**Example1:** Change the location and controlling department number of project number 10 to'Bellaire' and 5, respectively.

UPDATE PROJECT

SET PLOCATION = 'Bellaire', DNUM = 5 WHERE PNUMBER=10;

**Example2:** Give all employees in the 'Research' department a 10% raise in salary.



SSIT, Tumakuru Page 15

*DBMS Lab with mini project* *Dept. of CSE*

UPDATE EMPLOYEE

SET SALARY = SALARY \*1.1

WHERE DNO IN (SELECT DNUMBER FROM DEPARTMENT

WHERE DNAME='Research');

**1.9 VIEWS IN SQL**

|  |  |  |
| --- | --- | --- |
|  | A view is a single *virtual table* that is derived from other tables. The other tables could be |  |
|  | base tables or previously defined view. |  |
| Allows for limited update operations Since the table may not physically be stored |  |
|  |  |
| Allows full query operations |  |
|  |  |
| A convenience for expressing certain operations |  |
|  |  |
|  |  |

A view does not necessarily exist in physical form, which limits the possible update

* operations that can be applied to views.
* **Syntax: create view view\_name as**

**select columnnames from**

**tablename; Ex: create view emp\_info as**



 **select ename, address, salary from employee;**

**To display the details of above view give the following command Select \* from emp\_info;**



SSIT, Tumakuru Page 16

*DBMS Lab with mini project* *Dept. of CSE*

**LAB EXPERIMENTS**

*PART A: SQL PROGRAMMING*

1. **A. Consider the following schema for a Library Database:**

**BOOK (*Book\_id, Title, Publisher Name, Pub\_Year*)**

**BOOK\_AUTHORS (Book\_id, Author *Name*)**

**PUBLISHER (*Name, Address, Phone*)**

**BOOK\_COPIES (*Book\_id, Branch\_id, No-of Copies*)**

**BOOK\_LENDING (*Book\_id, Branch\_id, Card\_No, Date Out, Due Date*)**

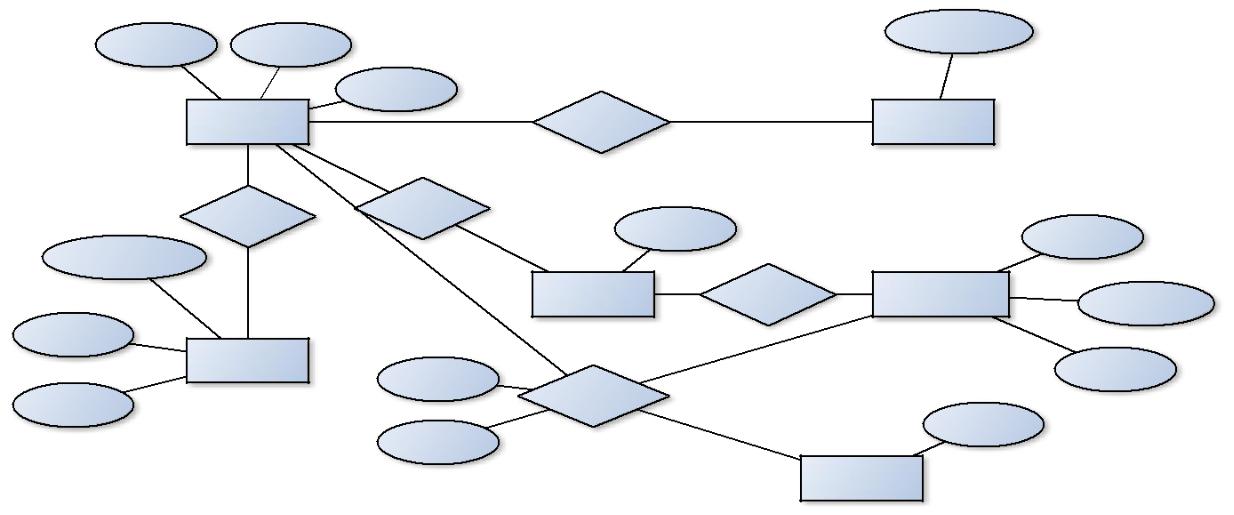
**LIBRARY\_BRANCH (*Branch\_id, Branch Name, Address*)**

**Write SQL queries to**

1. **Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.**
2. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017**
3. **Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.**
4. **Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.**
5. **Create a view of all books and its number of copies that are currently available in the Library.**

**Solution:**

**2.1 Entity-Relationship Diagram**



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **Book\_id** | | | Title |  |  |  |  |  |  | **Author\_Name** |  |  |  |  |
|  |  | Pub\_Year | M |  | N |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | Book |  |  | written-by |  |  | Book\_Authors | |  |  |  |  |
|  |  |  |  |  | N |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | Published-by | Has |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | N | No\_of\_copies |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | **Branch\_id** | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | **Publisher\_Name** | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 1 | M |  | M |  |  | N | |  |  |  |  |
|  |  |  |  |  | Book\_Copies | | In |  | Library\_Branch | |  |  |  |  |
|  |  |  |  |  |  |  |  | Branch\_Name | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Address | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | Publisher |  |  |  |  |  |  |  |  | Address | |  |
|  |  |  |  |  |  | Date\_out |  |  |  | N |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Phone | Book\_Lending |  |  |  |
|  |  |  |  |
|  |  |  | **Card\_No** |  |
|  | Due\_date |  |  |  |
|  | N | Card | |  |
|  |  |  |



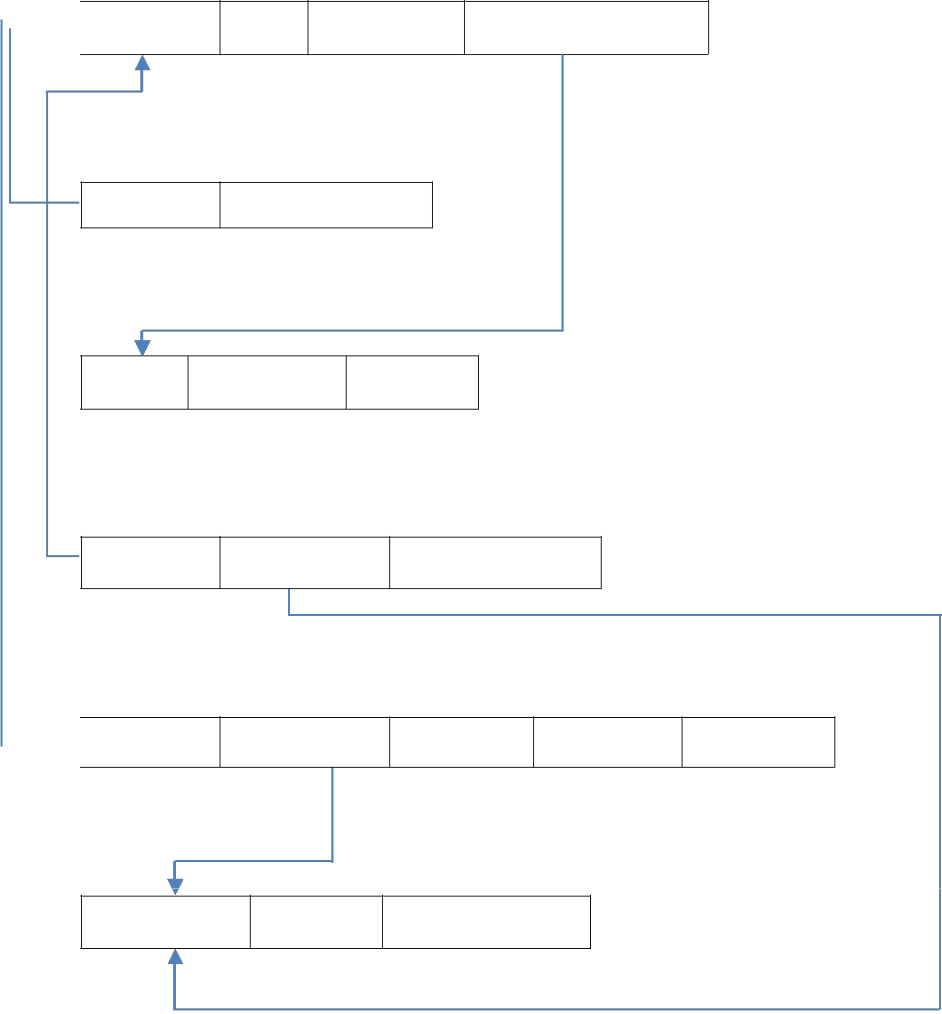
SSIT, Tumakuru Page 17

*DBMS Lab with mini project* *Dept. of CSE*

**2.1 Schema Diagram**

***Book***

 ***Book\_id*** *Title Pub\_Year Publisher\_Name*



***Book\_Authors***

***Book\_id Author\_name***

***Publisher***

***Name*** *Phone\_no* *Address*

***Book\_Copies***

***Book\_id*** ***Branch\_id*** *No\_of\_Copies*

***Book\_Lending***

 ***Book\_id Branch\_id Card\_no*** *Date\_out Due\_date*

***Library\_Branch***

***Branch\_id*** *Address* *Branch\_name*

**2.2 Table Creation**

SQL> create table publisher

(name varchar2(20),

address varchar(20),

phone number(10),

primary key(name));

Table created.

SQL> create table card



SSIT, Tumakuru Page 18

*DBMS Lab with mini project* *Dept. of CSE*

(card\_no number(10),

primary key(card\_no));

Table created.

SQL> create table library\_branch

(branch\_id number(10),

branch\_name varchar2(20),

address varchar2(20),

primary key(branch\_id));

Table created.

SQL> create table book

(book\_id number(10),

title varchar2(20),

publisher\_name varchar2(20),

pub\_year number(4),

primary key(book\_id),

foreign key(publisher\_name) references publisher(name)on delete cascade);

Table created.

SQL> create table book\_authors

(book\_id number(10),

authors\_name varchar2(20),

foreign key(book\_id) references book(book\_id)on delete cascade);

Table created.

SQL> create table book\_copies

(book\_id number(10),

branch\_id number(10),

no\_of\_copies number(10),

primary key(book\_id,branch\_id),

foreign key(book\_id) references book(book\_id)on delete cascade,

foreign key(branch\_id) references library\_branch(branch\_id)on delete cascade);

Table created.

SQL> create table book\_lending

(book\_id number(10),



SSIT, Tumakuru Page 19

*DBMS Lab with mini project* *Dept. of CSE*

branch\_id number(10),

card\_no number(10),

date\_out date,

due\_date date,

primary key(book\_id,branch\_id,card\_no),

foreign key(book\_id) references book(book\_id)on delete cascade, foreign key(branch\_id) references library\_branch(branch\_id)on delete cascade, foreign key(card\_no) references card(card\_no)on delete cascade);

|  |  |  |
| --- | --- | --- |
| Table created. |  |  |
| SQL> desc publisher; |  |  |
| Name | Null? | Type |
| ----------------------------------------- -------- ---------------------------- | | |
| NAME | NOT NULL VARCHAR2(20) | |
| ADDRESS |  | VARCHAR2(20) |
| PHONE |  | NUMBER(10) |
| SQL> desc card; |  |  |
| Name | Null? | Type |

----------------------------------------- -------------- ----------------------------

CARD\_NO NOT NULL NUMBER(10)

SQL> desc library\_branch;

Name Null? Type

----------------------------------------- ---------------- ----------------------------

BRANCH\_ID NOT NULL NUMBER(10)

BRANCH\_NAME VARCHAR2(20)

ADDRESS VARCHAR2(20)

SQL> desc book;

Name Null? Type

----------------------------------------- ------------- ----------------------------

BOOK\_ID NOT NULL NUMBER(10)

TITLE VARCHAR2(20)

PUBLISHER\_NAME VARCHAR2(20)

PUB\_YEAR NUMBER(4)



SSIT, Tumakuru Page 20

*DBMS Lab with mini project* *Dept. of CSE*

|  |  |  |
| --- | --- | --- |
| SQL> desc book\_authors; |  |  |
| Name | Null? | Type |
| ----------------------------------------- -------- ---------------------------- | | |
| BOOK\_ID |  | NUMBER(10) |
| AUTHORS\_NAME |  | VARCHAR2(20) |
| SQL> desc book\_copies; |  |  |
| Name | Null? | Type |

----------------------------------------- ------------- ----------------------------

|  |  |
| --- | --- |
| BOOK\_ID | NOT NULL NUMBER(10) |
| BRANCH\_ID | NOT NULL NUMBER(10) |
| NO\_OF\_COPIES | NUMBER(10) |
| SQL> desc book\_lending; |  |
| Name | Null?Type |

**----------------------------------------- ------------- ----------------------------**

BOOK\_ID NOT NULL NUMBER(10)

BRANCH\_ID NOT NULL NUMBER(10)

CARD\_NO NOT NULL NUMBER(10)

DATE\_OUT DATE

DUE\_DATE DATE

**2.2** Table insertion

SQL> insert into publisher values('pearson','delhi','8296463127'); 1 row created.

SQL> insert into publisher values('penglin','delhi','9606391794'); 1 row created.

SQL> insert into publisher values('john willy','mumbai','8105790195'); 1 row created.

SQL> insert into publisher values('eastman','mumbai','9105719519');



SSIT, Tumakuru Page 21

*DBMS Lab with mini project*

1 row created.

SQL> insert into publisher values('grow

hill','japan','9101002009'); 1 row created.

SQL> insert into card values('100');

1 row created.

SQL> insert into card values('101');

1 row created.

SQL> insert into card values('102');

1 row created.

SQL> insert into card values('103');

1 row created.

SQL> insert into card values('104');

1 row created.

SQL> insert into library\_branch

values('10','ssit','tumkur'); 1 row created.

SQL> insert into library\_branch

values('11','sit','tumkur'); 1 row created.

SQL> insert into library\_branch

values('12','cit','tumkur'); 1 row created.

SQL> insert into library\_branch

values('13','gm','mysore'); 1 row created.



*Dept. of CSE*

SSIT, Tumakuru Page 22

*DBMS Lab with mini project*

SQL> insert into library\_branch

values('14','bit','bangalore'); 1 row created.

SQL> insert into book

values('1','dbms','pearson','2017'); 1 row created.

SQL> insert into book

values('2','fafl','penglin','2012'); 1 row created.

SQL> insert into book values('3','os','john willy','2013'); 1 row created.

SQL> insert into book values('4','dc','eastman','2015'); 1 row created.

SQL> insert into book values('5','j2ee','grow hill','2013'); 1 row created.

SQL> insert into book\_authors

values('1','navathe'); 1 row created.

SQL> insert into book\_authors values('2','padhma reddy'); 1 row created.

SQL> insert into book\_authors values('3','behrow'); 1 row created.

SQL> insert into book\_authors

values('4','keogh'); 1 row created.

SQL> insert into book\_authors values('5','peter');



*Dept. of CSE*

SSIT, Tumakuru Page 23

*DBMS Lab with mini project*

1 row created.

SQL> insert into book\_copies

values('1','10','20'); 1 row created.

SQL> insert into book\_copies

values('2','11','30'); 1 row created.

SQL> insert into book\_copies

values('3','12','40'); 1 row created.

SQL> insert into book\_copies

values('4','13','50'); 1 row created.

SQL> insert into book\_copies

values('5','14','60'); 1 row created.

SQL> insert into book\_lending values('1','10','100','10-jan-2017','30-jun-2017'); 1 row created.

SQL> insert into book\_lending values('2','11','100','02-jan-2017','03-jun-2017'); 1 row created.

SQL> insert into book\_lending values('2','12','100','04-jan-2017','11-jun-2017'); 1 row created.

SQL> insert into book\_lending values('3','13','100','14-jan-2017','15-jun-2017'); 1 row created.

SQL> insert into book\_lending values('3','13','101','11-jan-2017','16-jun-2017'); 1 row created.



*Dept. of CSE*

SSIT, Tumakuru Page 24

*DBMS Lab with mini project* *Dept. of CSE*

|  |  |  |
| --- | --- | --- |
| SQL> select \* from publisher; | |  |
| NAME | ADDRESS | PHONE |
| ----------- | --------- | ---------- |
| pearson | delhi | 8296463127 |
| penglin | delhi | 9606391794 |
| john willy | mumbai | 8105790195 |
| eastman | mumbai | 9105719519 |
| grow hill | japan | 9101002009 |
| SQL> select \* from card; | |  |
| CARD\_NO |  |  |
| -------------------- | |  |
| 100 |  |  |
| 101 |  |  |
| 102 |  |  |
| 103 |  |  |
| 104 |  |  |

|  |  |  |
| --- | --- | --- |
| SQL> select \* from library\_branch; | |  |
| BRANCH\_ID | BRANCH\_NAME | ADDRESS |
| ---------- -------------------- ------------ | | ------------------- |
| 10 | ssit | tumkur |
| 11 | sit | tumkur |
| 12 | cit | tumkur |
| 13 | gm | mysore |
| 14 | bit | bangalore |

|  |  |  |  |
| --- | --- | --- | --- |
| SQL> select \* from book; | |  |  |
| BOOK\_ID | TITLE | PUBLISHER\_NAME | PUB\_YEAR |
| ---------- -------------------- | | -------------------- ---------- | -------------------- |
| 1 | dbms | pearson | 2017 |
| 2 | fafl | penglin | 2012 |
| 3 | os | john willy | 2013 |
| 4 | dc | eastman | 2015 |
| 5 | j2ee | grow hill | 2013 |

SQL> select \* from book\_authors;



SSIT, Tumakuru Page 25

|  |  |  |
| --- | --- | --- |
| *DBMS Lab with mini project* | | *Dept. of CSE* |
| BOOK\_ID AUTHORS\_NAME | |  |
| ---------- -------------------- | |  |
| 1 | navathe |  |
| 2 | padhma reddy |  |
| 3 | behrow |  |
| 4 | keogh |  |
| 5 | peter |  |

SQL> select \* from book\_copies;

|  |  |  |
| --- | --- | --- |
| BOOK\_ID BRANCH\_ID NO\_OF\_COPIES | | |
| ---------- ---------- | ------------ | ---------------------- |
| 1 | 10 | 20 |
| 2 | 11 | 30 |
| 3 | 12 | 40 |
| 4 | 13 | 50 |
| 5 | 14 | 60 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL> select \* from book\_lending; | | |  |  |
| BOOK\_ID BRANCH\_ID | | CARD\_NO | DATE\_OUT DUE\_DATE | |
| ---------- ---------- ---------- --------- --------- | | | ----------------- ---------------- | |
| 1 | 10 | 100 | 10-JAN-17 | 30-JUN-17 |
| 2 | 11 | 100 | 02-JAN-17 03-JUN-17 | |
| 2 | 12 | 100 | 04-JAN-17 | 11-JUN-17 |
| 3 | 13 | 100 | 14-JAN-17 | 15-JUN-17 |
| 3 | 13 | 101 | 11-JAN-17 | 16-JUN-17 |

**2.3** QUERIES

SQL> select

b.book\_id,b.title,b.publisher\_name,ba.authors\_name,bc.no\_of\_copies,l .branch\_id

1. from book b,book\_authors ba,book\_copies bc,library\_branch l
2. where b.book\_id=ba.book\_id and
3. b.book\_id=bc.book\_id and
4. bc.branch\_id=l.branch\_id;

|  |  |  |  |
| --- | --- | --- | --- |
| BOOK\_ID TITLE | | PUBLISHER\_NAMEAUTHORS\_NAME | |
| ---------- -------------------- | -------------------- | | -------------- ------------------------ |
| NO\_OF\_COPIES BRANCH\_ID | |  |  |
| ------------ ---------- ----------------- | |  |  |
| 1 dbms | pearson |  | Navathe |
| 20 | 10 |  |  |
|  |  |  |  |
|  |  |  |  |
| SSIT, Tumakuru |  |  | Page 26 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *DBMS Lab with mini project* | |  |  | *Dept. of CSE* |
| 2 fafl |  | penglin | padhma reddy |  |
| 30 | 11 |  |  |  |
| 3 os |  | john willy | behrow |  |
| 40 | 12 |  |  |  |
| BOOK\_ID TITLE | | PUBLISHER\_NAME | | AUTHORS\_NAME |
| ---------- -------------------- -------------------- ------------------- -------------------- | | | | |
| NO\_OF\_COPIES BRANCH\_ID | | |  |  |
| ------------ ---------- ----------------- | | |  |  |
| 4 dc |  | eastman | keogh |  |
| 50 | 13 |  |  |  |
| 5 j2ee |  | grow hill | peter |  |
| 60 | 14 |  |  |  |

SQL> select card\_no,count(\*)

1. from book\_lending
2. where date\_out between '01-jan-2017' and '30-jun-2017'
3. group by card\_no
4. having count(\*)>3;

|  |  |  |  |
| --- | --- | --- | --- |
| CARD\_NO | COUNT(\*) |  |  |
| ---------- ---------- | |  |  |
| 100 | 4 |  |  |
| SQL> select \* from book; | |  |  |
| BOOK\_ID TITLE | | PUBLISHER\_NAME | PUB\_YEAR |
| ---------- -------------------- -------------------- ---------- | | | ------------------------- |
| 1 dbms |  | pearson | 2017 |
| 2 fafl |  | penglin | 2012 |
| 3 os |  | john willy | 2013 |
| 4 dc |  | eastman | 2015 |
| 5 j2ee |  | grow hill | 2013 |
| SQL> select \* from book\_authors; | | |  |
| BOOK\_ID AUTHORS\_NAME | | |  |
| ---------- -------------------- | |  |  |
|  |  |  |  |
|  |  |  |  |
| SSIT, Tumakuru |  |  | Page 27 |

*DBMS Lab with mini project* *Dept. of CSE*

1. navathe
2. padhma reddy
3. behrow
4. keogh
5. peter

SQL> select \* from book\_copies;

|  |  |  |
| --- | --- | --- |
| BOOK\_ID BRANCH\_ID | | NO\_OF\_COPIES |
| ---------- ---------- | ------------ | ------------------------- |
| 1 | 10 | 20 |
| 2 | 11 | 30 |
| 3 | 12 | 40 |
| 4 | 13 | 50 |
| 5 | 14 | 60 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL> select \* from book\_lending; | | |  |  |
| BOOK\_ID BRANCH\_ID | | CARD\_NO | DATE\_OUT | DUE\_DATE |
| ---------- ---------- | ---------- --------- | --------- | ----------------- | ----------------- |
| 1 | 10 | 100 | 10-JAN-17 | 30-JUN-17 |
| 2 | 11 | 100 | 02-JAN-17 | 03-JUN-17 |
| 2 | 12 | 100 | 04-JAN-17 | 11-JUN-17 |
| 3 | 13 | 100 | 14-JAN-17 | 15-JUN-17 |
| 3 | 13 | 101 | 11-JAN-17 | 16-JUN-17 |

SQL> delete from book

1. where book\_id=1; 1 row delete

SQL> select \* from book;

BOOK\_ID TITLE PUBLISHER\_NAME PUB\_YEAR

---------- -------------------- -------------------- --------------- -----------------

2 fafl penglin 2012

3 os john willy 2013

4 dc eastman 2015

5 j2ee grow hill 2013

SQL> create view view\_publication as



SSIT, Tumakuru Page 28

*DBMS Lab with mini project* *Dept. of CSE*

1. select pub\_year from book; View created.

SQL> select \* from view\_publication;

PUB\_YEAR

----------

2012

2013

2015

2013

SQL> create view view\_book as

1. select b.book\_id,b.title,c.no\_of\_copies
2. from book b,book\_copies c,book\_lending l
3. where b.book\_id=c.book\_id and
4. c.branch\_id=l.branch\_id;

View created.

SQL> select \* from view\_book;

BOOK\_ID TITLE NO\_OF\_COPIES

---------- -------------------- ------------------------------

2 fafl 30

3 os 40

4 dc 50

4 dc 50



SSIT, Tumakuru Page 29

*DBMS Lab with mini project* *Dept. of CSE*

1. **B. Consider the following schema for order database: SALESMAN(Salesman\_id,Name,City,Commission) CUSTOMER(Customer\_id,Cust\_name,City,grade,Salesman\_id) ORDERS(Ord\_no,Purchase\_amt,Ord\_date,Customer\_id,Salesman\_id)**

**Write SQL queries to:**

**1.Count the customers with grade above banglore'saverage.**

**2.Find the name and numbers of all salesmen who had more than one customer.**

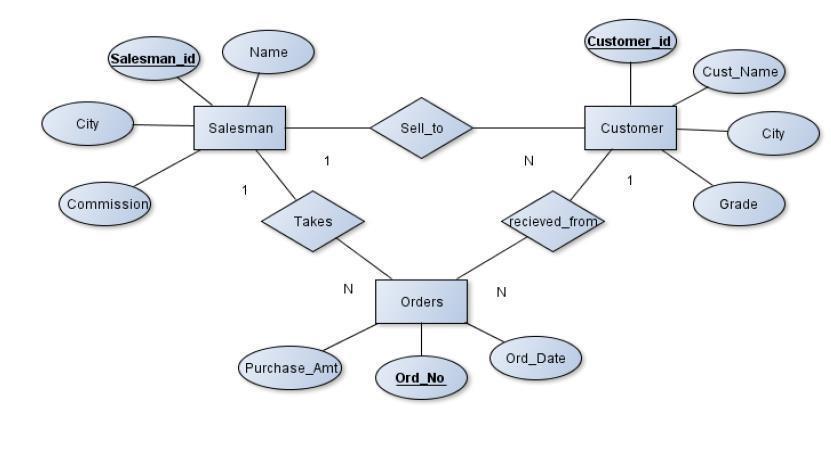
**3.List all salesman and indicate those who have and don't have customers in there cities (use UNION operation)**

**4.Create a view that finds the salesman who has the customer with the higest order of day.**

**5.Demonstrate the DELETE operation by removing salesman with id 1000.All his orders must also be deleted.**

**Solution:**

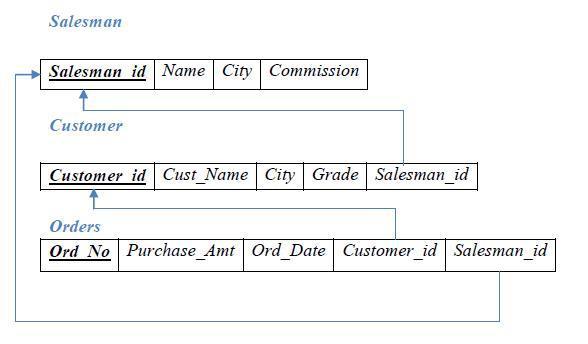
**3.1 Entity-Relationship Diagram**



SSIT, Tumakuru Page 30

*DBMS Lab with mini project* *Dept. of CSE*

**3.1** **Schema Diagram**



**3.2 Table Creation**

SQL> create table salesman

(salesman\_id number(10),

name varchar2(20),

city varchar2(20),

commission varchar2(10),

primary key(salesman\_id));

Table created.

SQL> create table customer

(customer\_id number(10),

cust\_name varchar2(20),

city varchar2(20),

grade number(5),

salesman\_id number(10),

primary key(customer\_id),

foreign key(salesman\_id) references salesman(salesman\_id)on delete set null);

Table created.

SQL> create table orders

(ord\_no number(10),

purchase\_amt number(10,2),

ord\_date date,



SSIT, Tumakuru Page 31



*DBMS Lab with mini project* *Dept. of CSE*

customer\_id number(10),

salesman\_id number(10),

primary key(ord\_no),

foreign key(customer\_id) references customer(customer\_id)on delete cascade, foreign key(salesman\_id) references salesman(salesman\_id)on delete cascade);

Table created.

SQL> desc salesman;

Name Null? Type

----------------------------------------- -------- ----------------------------

SALESMAN\_ID NOT NULL NUMBER(10)

NAME VARCHAR2(20)

CITY VARCHAR2(20)

COMMISSION VARCHAR2(10)

SQL> desc customer;

Name Null? Type

----------------------------------------- -------- ----------------------------

|  |  |
| --- | --- |
| CUSTOMER\_ID | NOT NULL NUMBER(10) |
| CUST\_NAME | VARCHAR2(20) |
| CITY | VARCHAR2(20) |
| GRADE | NUMBER(5) |
| SALESMAN\_ID | NUMBER(10) |
| SQL> desc orders; |  |
| Name | Null? Type |

----------------------------------------- -------- ----------------------------

ORD\_NO NOT NULL NUMBER(10)

PURCHASE\_AMT NUMBER(10,2)

ORD\_DATE DATE

CUSTOMER\_ID NUMBER(10)

SALESMAN\_ID NUMBER(10)

3.2 SQL> insert into salesman values('10','kavana','banglore','10%');

1 row created.

SQL> insert into salesman values('11','deepika','tumkur','20%');

1 row created.

SQL> insert into salesman values('12','chandu','hassan','30%');

1 row created.



SSIT, Tumakuru Page 32

*DBMS Lab with mini project*

SQL> insert into salesman

values('1000','keerthan','mandya','40%'); 1 row created.

SQL> insert into salesman

values('1001','anki','banglore','50%'); 1 row created.

SQL> insert into customer

values('123','bindu','banglore','2','10'); 1 row created.

SQL> insert into customer values('481','pranjali','tumkur','3','11'); 1 row created.

SQL> insert into customer values('432','kavya','durga','4','12'); 1 row created.

SQL> insert into customer values('111','vani','banglore','5','1000'); 1 row created.

SQL> insert into customer

values('112','shreya','banglore','6','1001'); 1 row created.

SQL> insert into customer

values('113','kruthi','banglore','6','1000'); 1 row created.

SQL> insert into customer values('114','havi','banglore','7','1001'); 1 row created.

SQL> insert into orders values('5','40','10-aug-2018','123','10'); 1 row created.

SQL> insert into orders values('6','50','11-aug-2018','481','11'); 1 row created.

SQL> insert into orders values('7','60','12-aug-2018','432','12'); 1 row created.



SSIT, Tumakuru

*Dept. of CSE*

Page 33

*DBMS Lab with mini project* *Dept. of CSE*

SQL> insert into orders values('8','70','13-aug-2018','111','1000');

1 row created.

SQL> insert into orders values('9','80','14-aug-2018','112','1001');

1 row created.

SQL> select \* from salesman;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SALESMAN\_ID | NAMECITY |  |  | COMMISSION | |
| ----------- ---------------- | ---------------- | | ---------- | |  |
| 10 | kavana | banglore | | 10% |  |
| 11 | deepika | tumkur | | 20% |  |
| 12 | chandu | hassan | | 30% |  |
| 1000 | keerthan | mandya | | 40% |  |
| 1001 | anki | banglore | | 50% |  |
| SQL> select \* from customer; | |  |  |  |  |
| CUSTOMER\_ID CUST\_NAME | |  | CITY | GRADE SALESMAN\_ID | |
| ----------- --------------- | ----------------- | | --- ---------- ----------------------------------- | | |
| 123 | bindu | banglore | | 2 | 10 |
| 481 | pranjali | tumkur | | 3 | 11 |
| 432 | kavya | durga | | 4 | 12 |
| 111 | vani | banglore | | 5 | 1000 |
| 112 | shreya | banglore | | 6 | 1001 |
| 113 | kruthi | banglore | | 6 | 1000 |
| 114 | havi | banglore | | 7 | 1001 |

7 rows selected.

SQL> select \* from orders;

ORD\_NO PURCHASE\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

---------- ------------ --------- ----------- -----------------------------------------------------

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 40 | 10-AUG-18 | 123 | 10 |
| 6 | 50 | 11-AUG-18 | 481 | 11 |
| 7 | 60 | 12-AUG-18 | 432 | 12 |
| 8 | 70 | 13-AUG-18 | 111 | 1000 |
| 9 | 80 | 14-AUG-18 | 112 | 1001 |

3.3 QUERIES

1. SQL> select grade,count(distinct customer\_id) 2 from customer

3 group by grade

4 having grade>(select avg(grade)



SSIT, Tumakuru Page 34

*DBMS Lab with mini project* *Dept. of CSE*

1. from customer
2. where city='banglore');

GRADE COUNT(DISTINCTCUSTOMER\_ID)

---------- --------------------------

|  |  |
| --- | --- |
| 6 | 1 |
| 5 | 1 |

2.

SQL> select salesman\_id,name

1. from salesman a
2. where 1<(select count(\*) from customer c
3. where c.salesman\_id=a.salesman\_id);

SALESMAN\_ID NAME

---------------------- ------

1000 keerthan

1001 anki

3.

SQL> (select s.salesman\_id,name,cust\_name,commission

1. from salesman s,customer c
2. where s.salesman\_id=c.salesman\_id and
3. s.city=c.city)
4. union
5. (select salesman\_id,name,'nomatch',commission
6. from salesman s
7. where not city=any(select city from customer))
8. order by 2 desc;

|  |  |  |  |
| --- | --- | --- | --- |
| SALESMAN\_ID | NAME | CUST\_NAME | COMMISSION |
| ----------------------- ----------- | | -------------------- | -------------------- |
| 1000 | keerthan | nomatch | 40% |
| 10 | kavana | bindu | 10% |
| 11 | deepika | pranjali | 20% |
| 12 | chandu | nomatch | 30% |
| 1001 | anki | havi | 50% |
| 1001 | anki | shreya | 50% |

6 rows selected.

4.

SQL> create view max\_salesman as

1. select b.ord\_date,a.salesman\_id,a.name
2. from salesman a,orders b
3. where a.salesman\_id=b.salesman\_id and
4. b.purchase\_amt=(select max(purchase\_amt)
5. from orders c
6. where b.ord\_date=c.ord\_date);



SSIT, Tumakuru Page 35

*DBMS Lab with mini project* *Dept. of CSE*

View created.

SQL> select \* from max\_salesman;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ORD\_DATE SALESMAN\_ID NAME | | |  |  |  |
| --------- ----------- ---------------------------- | | | |  |  |
| 10-AUG-18 | 10 | kavana |  |  |  |
| 11-AUG-18 | 11 | deepika |  |  |  |
| 12-AUG-18 | 12 | chandu |  |  |  |
| 13-AUG-18 | 1000 | keerthan | |  |  |
| 14-AUG-18 | 1001 | anki |  |  |  |
| SQL> select \* from salesman; | |  |  |  |  |
| SALESMAN\_ID | NAME | CITY | COMMISSION | | |
| ----------------------- | --------------- | ---------------- ------------------ | | |  |
| 10 | kavana | banglore | | 10% |  |
| 11 | deepika | tumkur | | 20% |  |
| 12 | chandu | hassan |  | 30% |  |
| 1000 | keerthan | mandya | | 40% |  |
| 1001 | anki | banglore | | 50% |  |
| SQL> select \* from customer; | |  |  |  |  |
| CUSTOMER\_ID | CUST\_NAME | | CITY | GRADE SALESMAN\_ID | |
| ------------------------- --------------------- ------------- | | |  | ----------- | ------------------------- |
| 123 | bindu | | banglore | 2 | 10 |
| 481 | pranjali | | tumkur | 3 | 11 |
| 432 | kavya | | durga | 4 | 12 |
| 111 | vani |  | banglore | 5 | 1000 |
| 112 | shreya | | banglore | 6 | 1001 |
| 113 | kruthi | | banglore | 6 | 1000 |
| 114 | havi | banglore | | 7 | 1001 |

7 rows selected.

SQL> select \* from orders;

ORD\_NO PURCHASE\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -------------- | ----------------------------- -------------- | | -------------------- | ----------------- |
| 5 | 40 | 10-AUG-18 | 123 | 10 |
| 6 | 50 | 11-AUG-18 | 481 | 11 |
| 7 | 60 | 12-AUG-18 | 432 | 12 |
| 8 | 70 | 13-AUG-18 | 111 | 1000 |
| 9 | 80 | 14-AUG-18 | 112 | 1001 |



SSIT, Tumakuru Page 36

|  |  |  |  |
| --- | --- | --- | --- |
| *DBMS Lab with mini project* | |  | *Dept. of CSE* |
|  | |  |  |
| SQL> delete from salesman  2 where salesman\_id='1000'; | |  |  |
| 1 row deleted. |  |  |  |
| SQL> select \* from salesman; | |  |  |
| SALESMAN\_ID | NAME | CITY | COMMISSION |
| --------------------- | ----------- | ------------ | ---------------------- |
| 10 | kavana | banglore | 10% |
| 11 | deepika | tumkur | 20% |
| 12 | chandu | hassan | 30% |
| 1001 | anki | banglore | 50% |





SSIT, Tumakuru Page 37

*DBMS Lab with mini project* *Dept. of CSE*

4

1. **Consider the schema foe the movie database: ACTOR(Act\_id,Act\_Name,Act\_Gender)**

**1. List the titles of all movies directed by 'Hitchcock'.**

**2.Find the movie names where one or more actors acted in two or more**

**movies.**

**3.List all actors who acted in a movie before 2000 and also in a movie after**

**2015(use join operation).**

**4.Find the title of movies and number of starts foe each movie that has atlest**

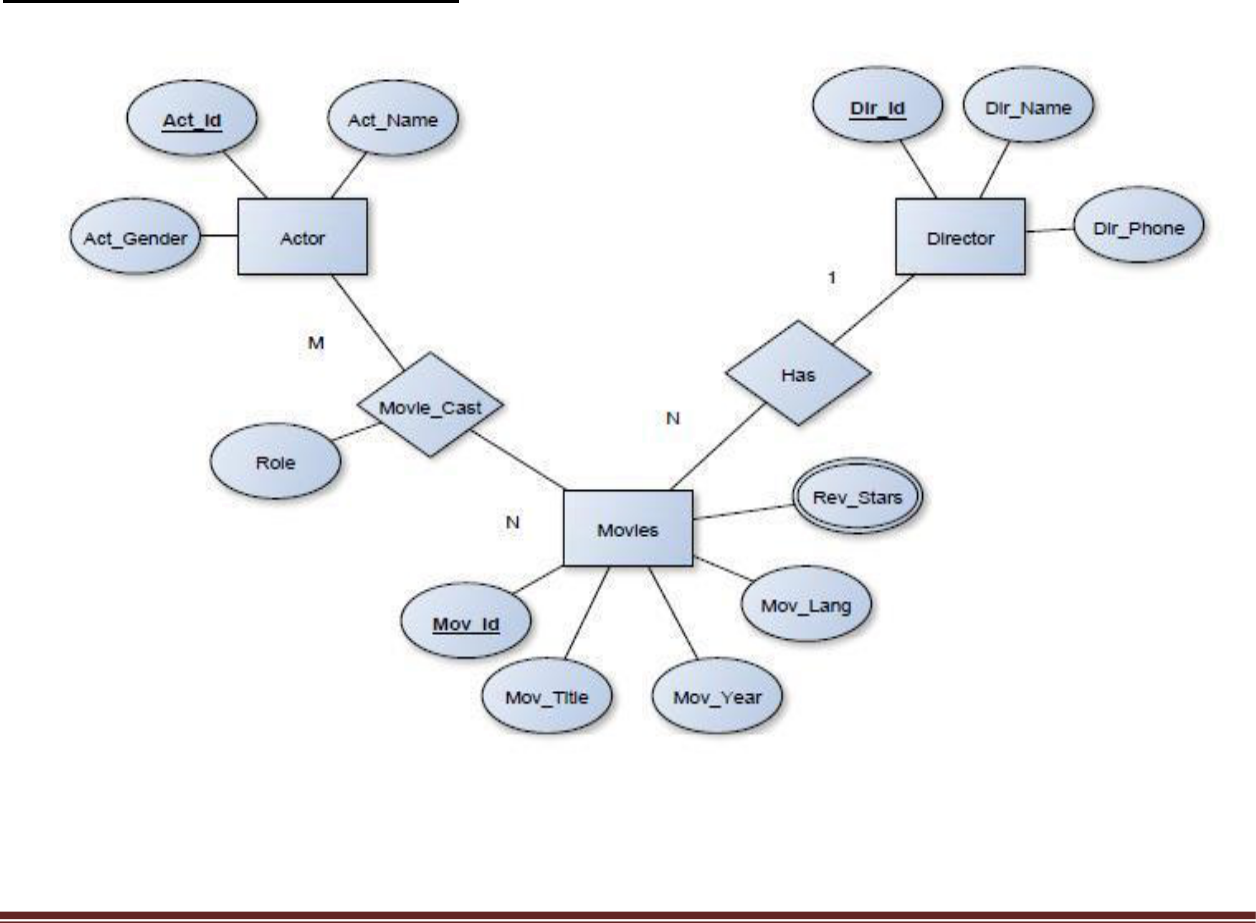
**one** **rating and find highest number of starts that movie recieved.Sort the result**

**by** **movie title.**

**5.Update rating of all movies directed by 'Steven Spielberg' to 5.**

**Solution:**

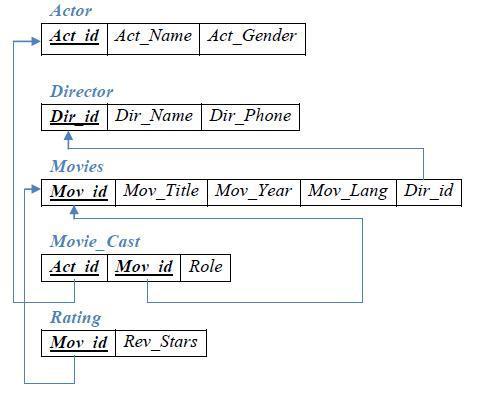
**4.1 Entity-Relationship Diagram**



SSIT, Tumakuru Page 38

*DBMS Lab with mini project*

**4.2 Schema Diagram**



4.3 **Table Creation**

SQL>create table actor

(act\_id number(10),

act\_name varchar2(10),

act\_gender char(1),

primary key(act\_id));

Table created.

SQL> create table director

(dir\_id number(10),

dir\_name varchar2(10),

dir\_phone number(10),

primary key(dir\_id));

Table created.

SQL>create table movies



SSIT, Tumakuru

*Dept. of CSE*

Page 39

*DBMS Lab with mini project* *Dept. of CSE*

(mov\_id number(10),

mov\_tittle varchar2(10),

mov\_year number(4),

mov\_lang varchar(10),

dir\_id number(10),

primary key(mov\_id),

foreign key(dir\_id) references director(dir\_id)on delete cascade);

Table created.

SQL>create table movie\_cast

(act\_id number(10),

mov\_id number(10),

role varchar(10),

primary key(act\_id,mov\_id),

foreign key(act\_id)references actor(act\_id)on delete cascade, foreign key(mov\_id)references movies(mov\_id)on delete cascade);

Table created.

SQL>create table rating

(mov\_id number(10),

rev\_stars number(4,2),

primary key(mov\_id),

foreign key(mov\_id)references movies(mov\_id)on delete cascade);

Table created.

SQL>desc actor;

Name Null? Type

----------------------------------------- -------- ----------------------------

ACT\_ID NOT NULL NUMBER(10)

ACT\_NAME VARCHAR2(10)

ACT\_GENDER CHAR(1)

SQL>desc director;

Name Null? Type

----------------------------------------- -------- ----------------------------

DIR\_ID NOT NULL NUMBER(10)

DIR\_NAME VARCHAR2(10)

DIR\_PHONE NUMBER(10)

SQL>desc movies;



SSIT, Tumakuru Page 40

*DBMS Lab with mini project*

*Dept. of CSE*

Name Null? Type ----------------------------------------- -------- ---------------------------- MOV\_ID NOT NULL NUMBER(10) MOV\_TITTLE VARCHAR2(10) MOV\_YEAR NUMBER(4) MOV\_LANG VARCHAR2(10) DIR\_ID NUMBER(10)

SQL>desc movie\_cast;

Name Null? Type

----------------------------------------- -------- ----------------------------

ACT\_ID NOT NULL NUMBER(10)

MOV\_ID NOT NULL NUMBER(10)

ROLE VARCHAR2(10)

SQL>desc rating;

Name Null? Type

----------------------------------------- -------- ----------------------------

MOV\_ID NOT NULL NUMBER(10)

REV\_STARS NUMBER(4,2) insert into actor

values(10,'ramya','f');

4.2

TABLE INSERTION

SQL> insert into actor values(10,'ramya','f');

1 row created.

SQL> insert into actor values(20,'radika','f');

1 row created.

SQL> insert into actor values(30,'punith','m');

1 row created.

SQL> insert into actor values(40,'punith','m');

1 row created.

SQL> insert into director values(50,'hitchcock',9964146221);

1 row created.

SQL> insert into director values(60,'sudeep',8296463127);



SSIT, Tumakuru Page 41

*DBMS Lab with mini project*

1 row created.

SQL> insert into director values(70,'prem',9606391794); 1 row created. SQL> insert into director values(80,'steven',8105790195); 1 row created.

SQL> insert into movies values(11,'akash','2014','kannada',50); 1 row created.

SQL> insert into movies

values(12,'drama','2013','kannada',60); 1 row created.

SQL> insert into movies

values(13,'ram','2008','kannada',70); 1 row created.

SQL> insert into movies values(14,'appu','1999','kannada',80); 1 row created.

SQL> insert into movie\_cast

values(10,11,'heroin'); 1 row created.

SQL> insert into movie\_cast

values(20,12,'heroin'); 1 row created.

SQL> insert into movie\_cast

values(30,13,'hero'); 1 row created.

SQL> insert into movie\_cast

values(40,14,'hero'); 1 row created.



*Dept. of CSE*

SSIT, Tumakuru Page 42

*DBMS Lab with mini project* *Dept. of CSE*

SQL> insert into movie\_cast

values(20,13,'villian'); 1 row created.

SQL> insert into movie\_cast

values(10,13,'villian'); 1 row created.

SQL> insert into movie\_cast

values(20,14,'villian'); 1 row created.

SQL> insert into rating values(11,5);

1 row created.

SQL> insert into rating values(12,4.5);

1 row created.

SQL> insert into rating values(13,3.2);

1 row created.

SQL> insert into rating values(14,4);

1 row created.

SQL> select \* from actor;

|  |  |  |
| --- | --- | --- |
| ACT\_ID ACT\_NAME | | A |
| ------------ | ------------ | ----- |
| 10 | ramya | f |
| 20 | radika | f |
| 30 | punith | m |
| 40 | punith | m |

SQL> select \* from director;

DIR\_ID DIR\_NAME DIR\_PHONE

--------- ----------------- ---------------



SSIT, Tumakuru Page 43

|  |  |  |
| --- | --- | --- |
| *DBMS Lab with mini project* | | *Dept. of CSE* |
| 50 | Hitchcock | 9964146221 |
| 60 | Sudeep | 8296463127 |
| 70 | Prem | 9606391794 |
| 80 | Steven | 8105790195 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SQL> select \* from movies; | | |  |  |  |
| MOV\_ID | MOV\_TITTLE MOV\_YEAR MOV\_LANG | | | | DIR\_ID |
| -------------- -------------------- | | | --------------- ------------------- | | ----------- |
| 11 | Akash | | 2014 | kannada | 50 |
| 12 | drama | | 2013 | kannada | 60 |
| 13 | Ram | | 2008 | kannada | 70 |
| 14 | Appu | | 1999 | kannada | 80 |
| SQL> select \* from movie\_cast; | | |  |  |  |
| ACT\_ID MOV\_ID ROLE | | |  |  |  |
| --------------- | ----------- | ------- |  |  |  |
| 10 | 11 | heroin |  |  |  |
| 20 | 12 | heroin |  |  |  |
| 30 | 13 | hero |  |  |  |
| 40 | 14 | hero |  |  |  |
| 20 | 13 | villian |  |  |  |
| 10 | 13 | villian |  |  |  |
| 20 | 14 | villian |  |  |  |
| SQL> select \* from rating; | | |  |  |  |
| MOV\_ID REV\_STARS | | |  |  |  |
| ---------------- | ---------------- | |  |  |  |
| 11 | 5 |  |  |  |  |
| 12 | 4.5 |  |  |  |  |
| 13 | 3.2 |  |  |  |  |
| 14 | 4 |  |  |  |  |

4.3 **QUERIES**

1.

SQL> select mov\_tittle

1. from movies
2. where dir\_id in(select dir\_id
3. from director
4. where dir\_name='hitchcock');



SSIT, Tumakuru Page 44

*DBMS Lab with mini project*

MOV\_TITTLE

----------

akash

2.

SQL> select mov\_tittle,count(\*)

2 from movies m,movie\_cast mc

1. where m.mov\_id=mc.mov\_id
2. and mc.act\_id in(select act\_id
3. from movie\_cast m
4. group by m.act\_id
5. having count(m.act\_id)>1) 8 group by mov\_tittle

9 having count(\*)>1;

MOV\_TITTLE COUNT(\*)

----------------- -------------

ram 2

3.

SQL> select a.act\_name,c.mov\_tittle,c.mov\_year 2 from actor a,movie\_cast b,movies c

1. where a.act\_id=b.act\_id and
2. b.mov\_id=c.mov\_id and
3. c.mov\_year not between 2000 and 2015;

ACT\_NAME MOV\_TITTLE MOV\_YEAR

---------------- ------------------- ----------------

radika appu 1999

punith appu 1999

4.

SQL> select mov\_tittle,max(rev\_stars)

from movies

inner join rating using(mov\_id)

group by mov\_tittle

having max(rev\_stars)>0

order by mov\_tittle;



SSIT, Tumakuru

*Dept. of CSE*

Page 45

*DBMS Lab with mini project* *Dept. of CSE*

MOV\_TITTLE MAX(REV\_STARS)

------------------- -----------------------------

akash 5

appu 5

drama 4.5

ram 3.2

5.

SQL> update rating

set rev\_stars=5

where mov\_id in(select mov\_id

from movies

where dir\_id in(select dir\_id

from director

where dir\_name='steven'));

1 row updated.

SQL> select \* from rating;

MOV\_ID REV\_STARS

--------------- ----------------

1. 5
2. 4.5
3. 3.2
4. 5



SSIT, Tumakuru Page 46

*DBMS Lab with mini project* *Dept. of CSE*

1. **D. Consider the schema for the company database: DEPARTMENT(DNo,DName,MgrSSN,MgrStartDate) EMPLOYEE(SSN,Name,Address,Sex,Salary,SuoerSSN,DNo) DLOCATION(DNo,DLoc) PROJECT(PNo,PName,PLoc,DNo)**

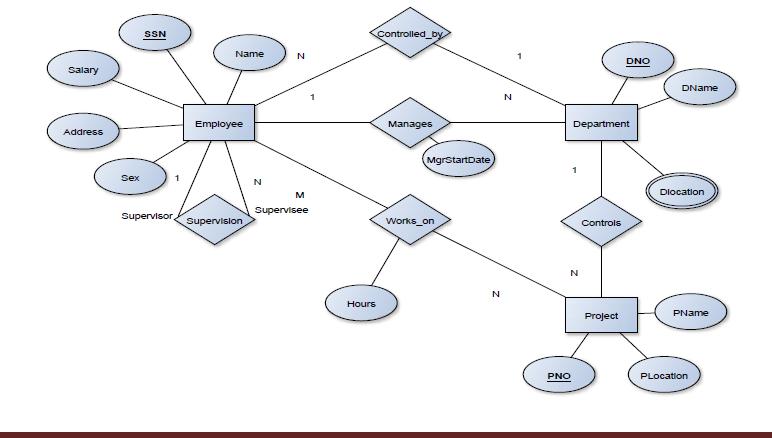
**WORKS\_ON(SSN,PNo,Hours)**

**Write SQL query to**

1. **Make alist of all project numbers for projects that involve an empolyee whose last name is 'Scott',either as a worker or a manager of the department that controls the project.**
2. **Show the resulting salaries if every employee works on the 'loT' project is given a 10% rise.**
3. **Find the sum of salaries of all employee of the 'Accounts' department as well as the maximum salary, the minimum salary, and the average salary in this department.**
4. **Retrieve the name of each employee who works on all the project controlled by department no 5(use NOT EXISTS operation)**
5. **For each department that has more than 5 employee,retrieve the department number and the no of its employee who are making more than Rs.6 lakh.**

**Solution:**

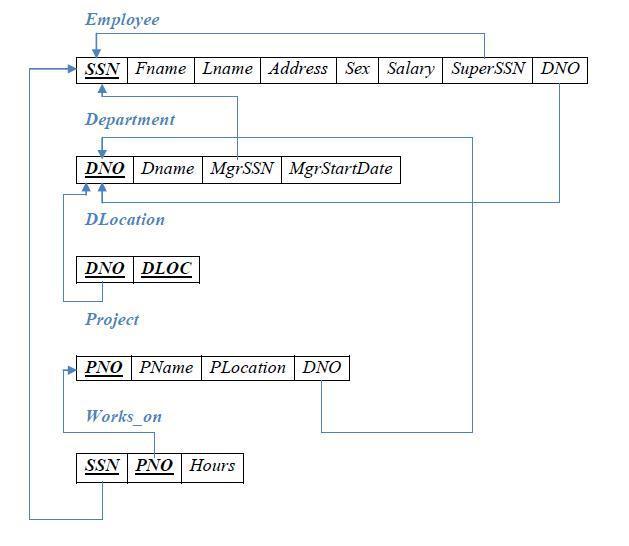
**5.1 Entity-Relationship Diagram**



SSIT, Tumakuru Page 47

*DBMS Lab with mini project* *Dept. of CSE*

**5.1 Schema Diagram**



**5.2Table Creation**

SQL>create table department

(dno varchar2(20),

dname varchar2(20),

mgrstartdate date,

primary key(dno));

Table created.



SSIT, Tumakuru Page 48

*DBMS Lab with mini project* *Dept. of CSE*

SQL>create table employee

(ssn varchar2(20),

name varchar2(20),

address varchar2(20),

sex char(1),

salary number(10,2),

superssn varchar2(20),

dno varchar2(20),

primary key(ssn),

foreign key(superssn)references employee(ssn)on delete cascade, foreign key(dno)references department(dno)on delete cascade);

Table created.

SQL>alter table department add mgrssn references employee(ssn);

Table altered.

SQL>create table dlocation

(dno varchar2(20),

dloc varchar2(20),

primary key(dno,dloc),

foreign key(dno)references department(dno)on delete cascade);

Table created.

SQL>create table project

(pno number(10),

pname varchar2(20),

ploc varchar2(20),

dno varchar2(20),

primary key(pno),

foreign key(dno)references department(dno)on delete cascade);

table created.

SQL>create table works\_on

(ssn varchar2(20),

pno number(10),

hours number(3,1),

primary key(ssn,pno),



SSIT, Tumakuru Page 49

*DBMS Lab with mini project* *Dept. of CSE*

foreign key(ssn)references employee(ssn)on delete cascade, foreign key(pno)references project(pno)on delete cascade);

Table created.

SQL> desc department;

Name Null? Type

----------------------------------------- -------- -------------------------

DNO NOT NULL NUMBER(10)

DNAME VARCHAR2(15)

MGRSTARTDATE DATE

MGRSSN NUMBER(10)

SQL> desc employee;

Name Null? Type

----------------------------------------- -------- -------------------------

|  |  |
| --- | --- |
| SSN | NOT NULL NUMBER(10) |
| NAME | VARCHAR2(15) |
| ADDRESS | VARCHAR2(15) |
| SEX | CHAR(1) |
| SALARY | NUMBER(10,2) |
| SUPERSSN | NUMBER(10) |
| DNO | NUMBER(10) |
| SQL> desc dlocation; |  |
| Name | Null? Type |

----------------------------------------- -------- -------------------------

DNO NOT NULL NUMBER(10)

DLOC NOT NULL VARCHAR2(15)

SQL> desc project;

Name Null? Type

----------------------------------------- -------- -------------------------

PNO NOT NULL NUMBER(10)

PNAME VARCHAR2(15)

PLOC VARCHAR2(15)

DNO NUMBER(10)

SQL> desc works\_on;

Name Null? Type



SSIT, Tumakuru Page 50

|  |  |
| --- | --- |
| *DBMS Lab with mini project* | *Dept. of CSE* |
| ------------------------------------ -------- ------------------------- | |
| SSN | NOT NULL NUMBER(10) |
| PNO | NOT NULL NUMBER(10) |
| HOURS | NUMBER(3,1) |

**5.3 TABLE INSERTION**

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs01','scott','banglore','m',450000);

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs02','smith','banglore','m',500000);

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs03','baker','banglore','m',700000);

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs04','satt','manglore','m',500000);

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs05','pavan','manglore','m',650000);

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs06','neha','banglore','f',800000);

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs07','ahana','mysore','f',350000);

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs08','veena','mysore','f',600000);



SSIT, Tumakuru Page 51

*DBMS Lab with mini project* *Dept. of CSE*

1 row created.

SQL>insert into

employee(ssn,name,address,sex,salary)values('16cs09','nagesh','tumkur','m',500000);

1 row created.

SQL>insert into department values(1,'accounts','01-jan-2001','16cs07'); 1 row created.

SQL> insert into department values(2,'it','01-aug-2006','16cs09'); 1 row created.

SQL>insert into department values(3,'ece','01-jun-2008','16cs01'); 1 row created.

SQL>insert into department values(4,'ise','01-aug-2015','16cs08'); 1 row created.

SQL>insert into department values(5,'cse','01-jun-2002','16cs05');

SQL> update employee set superssn=null,dno=3

where ssn='16cs01';

1 row updated.

SQL>update employee set

superssn='16cs01',dno=5 where ssn='16cs01';

1 row updated.

SQL> update employee set

superssn='16cs04',dno=5 where ssn='16cs04';

1 row updated.

SQL> update employee set superssn='16cs03',dno=5



SSIT, Tumakuru Page 52

*DBMS Lab with mini project* *Dept. of CSE*

where ssn='16cs03';

1 row updated.

SQL>update employee set

superssn='16cs02',dno=5 where ssn='16cs02';

1 row updated.

SQL>update employee set

superssn='16cs05',dno=5 where ssn='16cs05';

1 row updated.

SQL>update employee set

superssn='16cs06',dno=1 where ssn='16cs06';

1 row updated.

SQL>update employee set superssn=null,dno=2

where ssn='16cs07';

1 row updated.

SQL>update employee set superssn=null,dno=4

where ssn='16cs08';

1 row updated.

SQL>update employee set superssn=null,dno=2

where ssn='16cs09';

1 row updated.

SQL>insert into dlocation values(1,'banglore');

1 row created.

SQL>insert into dlocation values(2,'banglore');

1 row created.



SSIT, Tumakuru Page 53

*DBMS Lab with mini project*

SQL>insert into dlocation values(3,'banglore'); 1 row created.

SQL>insert into dlocation values(4,'manglore'); 1 row created.

SQL>insert into dlocation values(5,'manglore');

1 row created.

SQL>insert into project

values(100,'iot','banglore',5); 1 row created.

SQL>insert into project

values(101,'cloud','banglore',5); 1 row created.

SQL>insert into project

values(102,'bigdata','banglore',5); 1 row created.

SQL>insert into project values(103,'sensor','mysore',3);

1 row created.

SQL>insert into project values(104,'bank','banglore',1); 1 row created.

SQL>insert into project values(105,'salarymanagement','banglore',1); 1 row created.

SQL>insert into project values(106,'open stack','banglore',4); 1 row created.

SQL>insert into project values(107,'smart city','banglore',2);



*Dept. of CSE*

SSIT, Tumakuru Page 54

*DBMS Lab with mini project*

1 row created.

SQL>insert into works\_on

values('16cs01',100,4); 1 row created.

SQL>insert into works\_on

values('16cs01',101,3); 1 row created.

SQL>insert into works\_on

values('16cs01',102,8); 1 row created.

SQL>insert into works\_on

values('16cs02',101,4); 1 row created.

SQL> insert into works\_on

values('16cs04',105,6); 1 row created.

SQL>insert into works\_on

values('16cs05',106,5); 1 row created.

SQL>insert into works\_on

values('16cs03',107,5); 1 row created.

SQL>insert into works\_on

values('16cs01',109,6); 1 row created.

SQL>insert into works\_on

values('16cs07',107,7); 1 row created.

SQL>insert into works\_on

values('16cs08',101,5); 1 row created.



*Dept. of CSE*

SSIT, Tumakuru Page 55

*DBMS Lab with mini project* *Dept. of CSE*

SQL>insert into works\_on values('16cs09',102,8);

1 row created.

SQL>insert into works\_on values('16cs09',103,4);

1 row created.

SQL> select \* from employee;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SSN | NAME | ADDRESS |  | S | SALARY |
| -------------------- -------------------- -------------------- - ---------- | | | | | |
| SUPERSSN | DNO |  |  |  |  |
| -------------------- -------------------- | | |  |  |  |
| 16cs01 | scott | banglore | m | 450000 | |
| 16cs01 | 5 |  |  |  |  |
| 16cs02 | smith | banglore | m | 500000 | |
| 16cs02 | 5 |  |  |  |  |
| 16cs03 | baker | banglore | m | 700000 | |
| 16cs03 | 5 |  |  |  |  |
| SSN | NAME | ADDRESS |  |  | SALARY |
| -------------------- -------------------- -------------------- - ---------- | | | | | |
| SUPERSSN | DNO |  |  |  |  |
| -------------------- -------------------- | | |  |  |  |
| 16cs04 | satt | manglore | m | 500000 | |
| 16cs04 | 5 |  |  |  |  |
| 16cs05 | pavan | manglore | m | 650000 | |
| 16cs05 | 5 |  |  |  |  |
| 16cs06 | neha | banglore | f | 800000 | |
| 16cs06 | 1 |  |  |  |  |
| SSN | NAME | ADDRESS |  | S | SALARY |
| -------------------- -------------------- -------------------- - ---------- | | | | | |
| SUPERSSN | DNO |  |  |  |  |
| -------------------- -------------------- | | |  |  |  |



SSIT, Tumakuru Page 56

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *DBMS Lab with mini project* | |  |  | *Dept. of CSE* |
| 16cs07 | ahana | mysore | f | 350000 |
|  | 2 |  |  |  |
| 16cs08 | veena | mysore | f | 600000 |
|  | 4 |  |  |  |
| 16cs09 | nagesh | tumkur | m | 500000 |
|  | 2 |  |  |  |

9 rows selected.

SQL> select \* from department;

|  |  |  |  |
| --- | --- | --- | --- |
| DNO | DNAME | MGRSTARTD MGRSSN | |
| -------------------- ------------------ | | -- --------- ------ -------------- | |
| 1 | accounts | 01-JAN-01 | 16cs07 |
| 2 | it | 01-AUG-06 | 16cs09 |
| 3 | ece | 01-JUN-08 | 16cs01 |
| 4 | ise | 01-AUG-15 | 16cs08 |
| 5 | cse | 01-JUN-02 | 16cs05 |
| SQL> select \* from dlocation; | |  |  |
| DNO | DLOC |  |  |
| ----------------------------- | |  |  |
| 1 | banglore |  |  |
| 2 | banglore |  |  |
| 3 | banglore |  |  |
| 4 | manglore |  |  |
| 5 | manglore |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| SQL> select \* from project; | |  |  |
| PNO | PNAME | PLOC | DNO |
| ---------- -------------------- -------------------- -------------------- | | | |
| 100 | iot | banglore | 5 |
| 101 | cloud | banglore | 5 |
| 102 | bigdata | banglore | 5 |
| 103 | sensor | mysore | 3 |
| 104 | bank | banglore | 1 |
| 105 | salarymanagement | banglore | 1 |
|  |  |  |  |
|  |  |  |  |
| SSIT, Tumakuru |  |  | Page 57 |

|  |  |  |  |
| --- | --- | --- | --- |
| *DBMS Lab with mini project* | |  | *Dept. of CSE* |
| 106 | open stack | banglore | 4 |
| 107 | smart city | banglore | 2 |

8 rows selected.

SQL> select \* from works\_on;

|  |  |  |
| --- | --- | --- |
| SSN | PNO | HOURS |
| -------------------- ---------- ---------- | | |
| 16cs01 | 100 | 4 |
| 16cs01 | 101 | 3 |
| 16cs01 | 102 | 8 |
| 16cs02 | 101 | 4 |
| 16cs04 | 105 | 6 |
| 16cs05 | 106 | 5 |
| 16cs03 | 107 | 5 |
| 16cs07 | 107 | 7 |
| 16cs08 | 101 | 5 |
| 16cs09 | 102 | 8 |
| 16cs09 | 103 | 4 |

11 rows selected.

**5.4 QUERIES** 1.

SQL> (select distinct p.pno

2 from project p,department d,employee e

3 where e.dno=d.dno and

4 d.mgrssn=e.ssn and

1. e.name='scott')
2. union

7 (select distinct p1.pno

8 from project p1,works\_on w,employee e1

9 where p1.pno=w.pno and

10 e1.ssn=w.ssn and

11 e1.name='scott');

PNO

----------

100

101

102



SSIT, Tumakuru

Page 58

*DBMS Lab with mini project* *Dept. of CSE*

2.

SQL> select e.name,e.salary\*1.1 as "inc\_sal"

2 from employee e,works\_on w,project p

3 where e.ssn=w.ssn and

4 w.pno=p.pno and

5 p.pname='iot';

NAME inc\_sal

-------------------- ----------

scott 495000

3.

SQL> select sum(e.salary),max(salary),min(salary),avg(salary)

2 from employee e,department d

3 where e.dno=d.dno and

4 d.dname='accounts';

SUM(E.SALARY) MAX(SALARY) MIN(SALARY) AVG(SALARY)

------------- ----------- ----------- -----------

800000 800000 800000 800000

SQL> update department set

dname='accounts'

where dno=2;

1 row updated.

SQL> select sum(e.salary),max(salary),min(salary),avg(salary)

2 from employee e,department d

3 where e.dno=d.dno and

4 d.dname='accounts';

SUM(E.SALARY) MAX(SALARY ) MIN(SALARY) AVG(SALARY)

------------- ----------- ----------- -------------------------------------------------------

1650000 800000 350000 550000

4.

SQL> select e.name

2 from employee e

3 where not exists ((select pno

4 from project

5 where dno=5)



SSIT, Tumakuru Page 59

1. minus (select pno
2. from works\_on
3. where e.ssn=ssn));

NAME

--------------------

scott

5.

SQL> select d.dno,count(\*)

2 from department d,employee e

3 where d.dno=e.dno and

4 salary>600000 and

5 d.dno in (select e.dno

6 from employee e

7 group by e.dno

8 having count(\*)>5)

9 group by d.dno;

no rows selected

SQL> update employee set

superssn=null,dno=5

where ssn='16cs06';

1 row updated.

SQL> select d.dno,count(\*)

2 from department d,employee e

3 where d.dno=e.dno and

4 salary>600000 and

5 d.dno in (select e.dno

6 from employee e

7 group by e.dno

8 having count(\*)>5)

9 group by d.dno;

DNO COUNT(\*)

-------------------- ----------

5 3





**SSIT, Tumakuru Page 60**

PART B

HOTEL

MANAGEMENT

SYSTEM

**PART -B**

**TABLE CONTENT**

|  |  |  |
| --- | --- | --- |
| **Sr no** | **Tittle** | **Page no** |
| **1.** | **Introduction and Objectives** | **66** |
| **1.1** | **Problem statement** |  |
| **1.2** | **Objective** | **67** |
| **1.3** | **Advantages and Disadvantages** |  |
| **2.** | **Project management** | **68** |
| **2.1** | **Project planning & scheduling** |  |
| **2.1.1** | **Methodology** |  |
| **2.1.2** | **Life cycle** | **69** |
| **2.1.3** | **Project plan** |  |
| **2.2** | **Risk management** | **70** |
| **3** | **System analysis** | **71** |
| **3.1** | **Background study** |  |
| **3.2** | **Software system** |  |
| **3.4** | **Feasibility study** | **72** |
| **4** | **System design** | **73** |
| **4.1** | **Data design** |  |
| **4.2** | **Er diagram** | **74** |
| **4.3** | **Discussion of Result** | **75-79** |
| **Sr no** | **Title** | **Page no** |
| **5** | **Implementation** | **80-82** |
| **6** | **PHP-code** | **83-89** |
| **7** | **Conclusion** | **90** |
| **7.4** | **Refences** | **91** |

**Chapter One**

**1.Introduction**

Hotel management systems were first introduced in the late 1970s as relatively simple standalone

systems to control room inventory, maintain room booking schedules, assignment. The modern hotel management system is comprehensive suite of products to provide a system that assists with avariety of room management tasks and service customer needs from the time of initial reservation through completion of the check-out.

A Hotel Management system (HMS) is part of the so-called customer service systems (CSS),

which are applications supporting the direct contact with the customer. HMS eventually evolved into the computer reservations system (CRS).

**1.1 Problem Statement**

The project, Hotel Management System is a web-based application that allows the hotel manager

to handle all hotel activities online. Interactive GUI and the ability to manage various room bookings make this system very flexible and convenient. The hotel manager is a very busy person and does not have the time to sit and manage the entire activities manually on paper. This application gives him the power and flexibility to manage the entire system from a single online system. Hotel management project provides room booking, staff management and other necessary hotel management features. The system allows the manager to post available rooms in the system. Customers can view and book room online. Admin has the power of either approving or disapproving the customer’s booking request. Other hotel services can also be viewed by the customers and can book them too. The system is hence useful for both customers and managers to portable manage the hotel activities

**1.2 Objective**

The main objective of the entire activity is to automate the process of day to day activities

of Hotel like:

* Room activities.
* Admission of new customer.
* Assign a room according to customer’s demand.
* Checkout of a computer and releasing the room
* Finally copute the bill.
* Packages available.
* Admin booking confirmation.
* List of customers who booked.
* News letter.
* Hotel facilities.
* Check-in and check-out of customer.
* Invoice of the cutomer.

This project is used bye two types of users

* Online users.
* Administrator(Manager of the Hotel).

Online users can see rooms types and the facilites that are provided by the hotel.

Administrator can maintain daily updates in the hotel records.Administrator must

be an authorized user.He can further change password.There is the facility for password

recovery,logout etc.

“The HOTEL maintains four different type of payment viz. Cash payment, DD payment, CREDIT Card payment. Every transaction has to pass through any one of these payment”.

There are four different types of rooms superior room, Deluxe room, guest room and single room

customer can choose any type of room and also have Meals plan and Bedding type.

**Hotel tag line:** “you’re not lucky to get best services, you worth it”.

**1.3.1 Advantages**

* Sometime it happens that the rooms get booked soon when one visits the place therefore user can make advance booking using this system.
* It saves user time in searching a room.
* The system is useful as it calculates an exact cost of rooms for requested number of days.
* It saves organization resources and expenses.
* This system is effective and saves time and cost of users.

**1.3.2 Disadvantages**

* The booking process usually requires a customer identity, which the system cannot detect.
* It requires a reliable internet connection.

**Chapter Two**

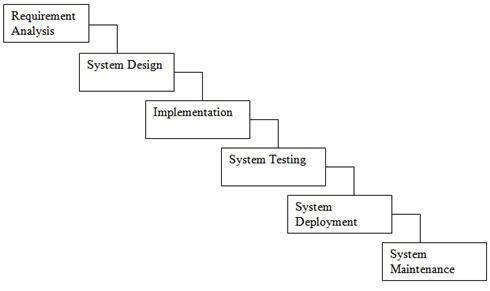
**Project Management**

**2.1 Project planning and scheduling**

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path.

**2.1.1 Methodology**

We have used Iterative and Incremental Development model (IID) for our project development. This development approach is also referred to as Iterative Waterfall Development approach. Iterative and Incremental Development is a software development process developed in response to the more traditional waterfall model. This model is designed to take care of such big project. The large and complicate project chiefly demand better development and testing procedure. The waterfall model is well known for its repeated testing process. Hence, I choose the waterfall model for developing my software.



**Fig. 2.1:** Waterfall model

Some advantages of waterfall model:

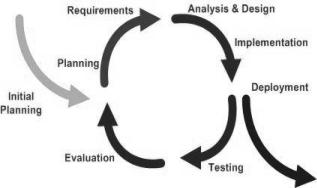
1. Simple and easy to understand and use.
2. Easy to manage due to the rigidity of the model.
3. Phases are processed and completed one at a time.
4. Works well for smaller projects where requirements are very well understood.

**2.1.2 Project Management Life Cycle**

The Project Management Life Cycle has four phases. Each project life cycle phase is described along with the tasks need to complete it

The four phases is

1. Initiation
2. Planning
3. Execution
4. Closure.



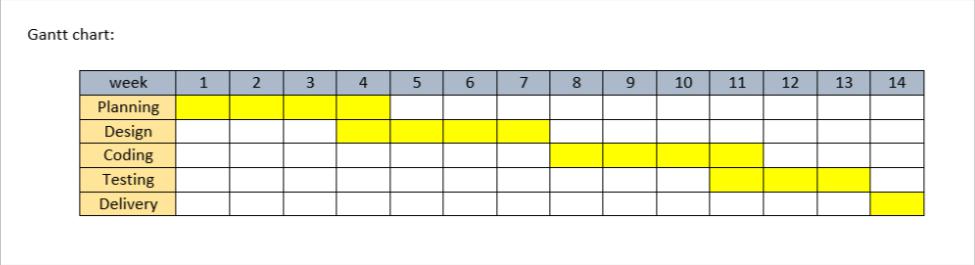
**Fig. 2.2:** Iterative and Incremental Life Cycle

**2.1.3 Project Plan:**

Once we examine that the project is feasible, I undertake project planning. The table below describes how we planned my project

**2.1.4 Schedule Representation**

Scheduling the project tasks is an important project planning activity. It involves deciding which tasks would be taken up when. In order to schedule the project activities, a software project manager needs to do the following this rules.



**Fig. 2.3:** Gantt chart

**2.2 Risk Management**

Software Risk Management is a proactive approach for minimizing the uncertainty and potential loss associated with a project. Some categories of risk include product size, business impact, customer-related, process, technology, development environment, staffing (size and experience), schedule, and cost. Risk Management is a practice with processes, methods, and tools for managing risks in a project.

Risk identification is a systematic attempt to specify threats to the project plan. By identifying known and predictable risks, we can take a first step toward avoiding them when possible and controlling them when necessary. To perform the risk identification, we categorized the risk into different categories as:

1. Technical Risk
2. Business Risk
3. Known Risk
4. Predictable Risk
5. Project Risk
6. Unpredictable

**Chapter Three**

**System Analysis**

**3.1 Background Study**

System Analysis is a separation of a substance into parts for study and their implementation and detailed examination.

Before designing any system it is important that the nature of the business and the way it currently operates are clearly understood. The detailed examination provides the specific data required during designing in order to ensure that all the client's requirements are fulfilled. The investigation or the study conducted during the analysis phase is largely based on the feasibility study. Rather it would not be wrong to say that the analysis and feasibility phases overlap. High-level analysis begins during the feasibility study. Though analysis is represented as one phase of the system development life cycle (SDLC), this is not true. Analysis begins with system initialization and continues until its maintenance. Even after successful implementation of the system, analysis may play its role for periodic maintenance and up gradation of the system. One of the main causes of project failures is inadequate understanding, and one of the main causes of inadequate understanding of the requirements is the poor planning of system analysis.

**3.2 Software system attributes**

**3.2.1 Reliability:** This application is a reliable product that produces fast & verified output of all its

process.

**3.2.2 Availability:** This application will be available to use and help them to carry their operations

conveniently.

**3.2.3 Security:** This application will be designed in a maintainable manner. It will be easy toincorporate new requirements in the individual modules.

**3.4 Feasibility study**

**3.4.1 Technical Feasibility:** This is concerned with specifying equipment and software that willsuccessfully satisfy the user requirement; the technical needs of the system may vary considerably, but might include:

The facility to produce outputs in a given time:

1. Response time under conditions.
2. Ability to process a certain volume of transaction at a particular seep.
3. Facility to communicate data to distant location.

**3.4.2 Operational Feasibility:** It is mainly related to human organization and political aspects. Thepoints to be considered are:

1. What changes will be brought with the system?
2. What organizational structures are distributed?
3. What new skills will be required? Do the existing staff members have these skills? If not, can then the trained due course of time

**3.4.3 Economic Feasibility:** Economic analysis is the most frequently used technique for evaluatingthe effectiveness of a proposed system. More frequently known as cost/benefit system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system.

**3.4.4 Management Feasibility:** It is a determination of whether a proposed project will beacceptable to management. If does not accept a project of gives a negligible support to it; the analyst will tend to view the project as a no feasible one.

**3.4.5 Social Feasibility:** Social feasibility is a determination of whether the project will beacceptable to the people or not. This determination typically examines the probability of the project accepted by the group directly affected by the proposed system change.

**Chapter Four**

**System Design**

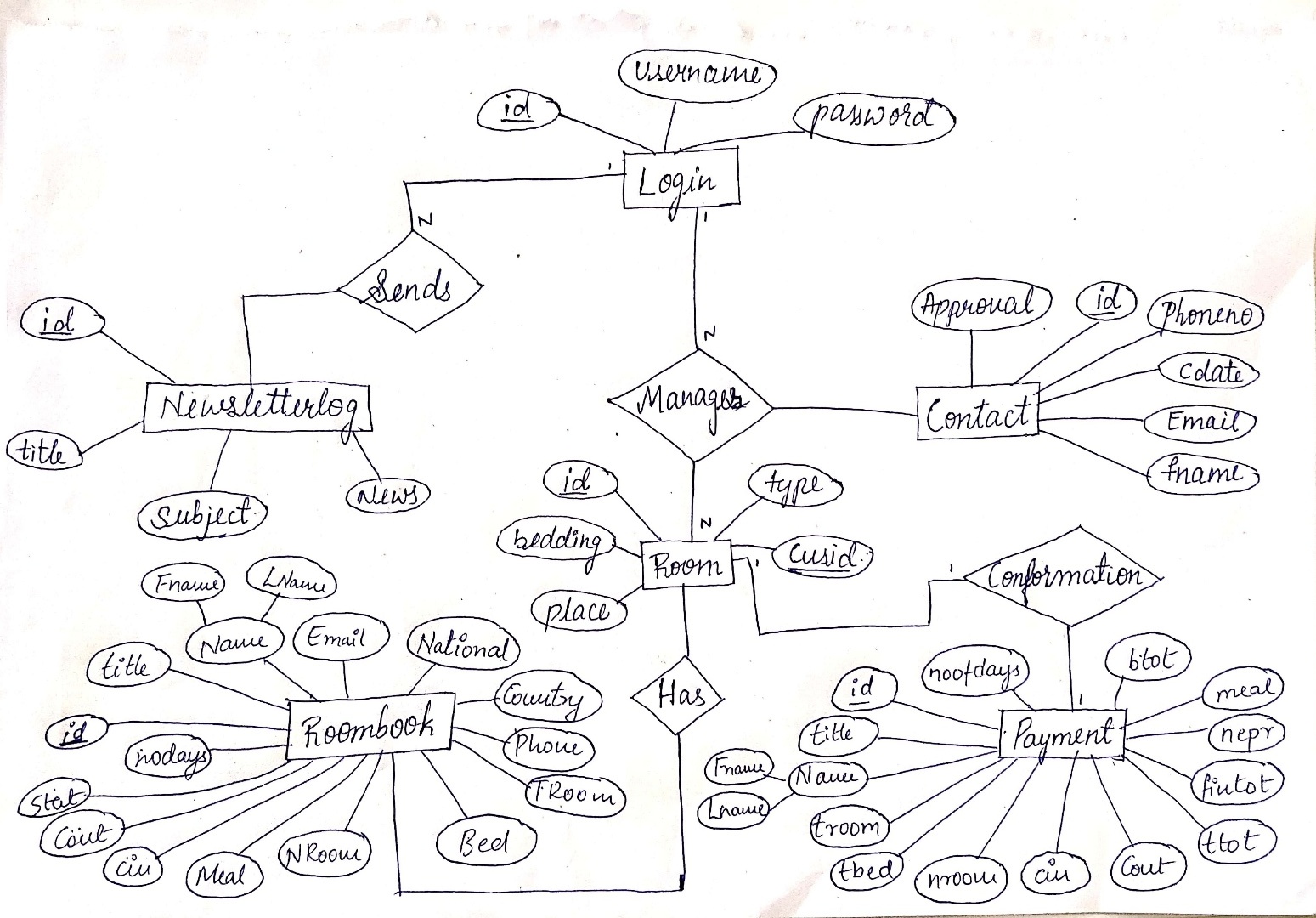
**4.1 Database Design**

Database design is the process of producing a detailed data model of database. This data model contains all the necessary logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structure used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structure, but also the forms and queries used as part of the overall database application within the database management system.

**4.2 E-R Diagram of Hotel Management System**

An entity-relationship diagram (ERD) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.



**Fig. 4.2:** E-R Diagram of Hotel Management System.

**1.Contact:** This table has id, fullname, phoneno, email, cdate, approval attributes .

where id(cust\_id )is the primary key. It is signup for Newsletters.

**2.Login:** This login table contain attributes like id, username and password.

where id(admin\_id) is the primary key. It is for admin Login.

**3.Newsletterlog:** This table contains attributes like id, title, subject, news.

where id(cust\_id) is the primary key.

**4.Payment:** This payment table contains numerous attributes like id, title, fname, lname, troom, tbed,

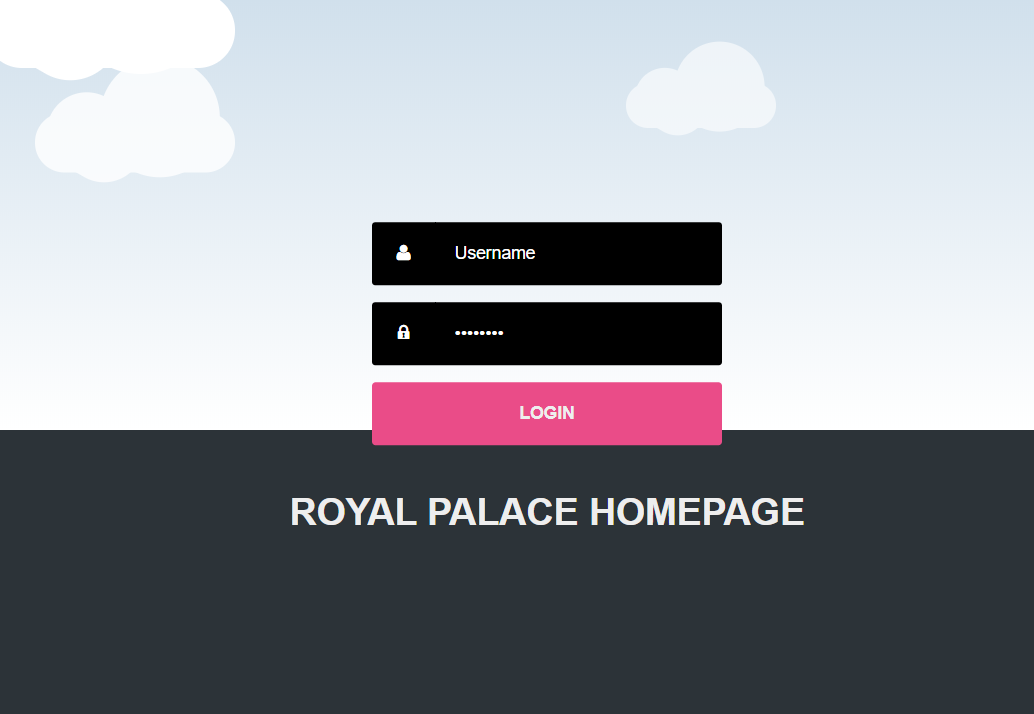
nroom, cin, cout, ttot, fintot, mepr, meal, btot, noofdays.

**5.Room:** This table has attributes like id, title, fname, lname, Email, National, country, phone, Troom,

Bed, Nroom, Meal, cin, cout, stat, nodays. where id(cust\_id)is the primary key.

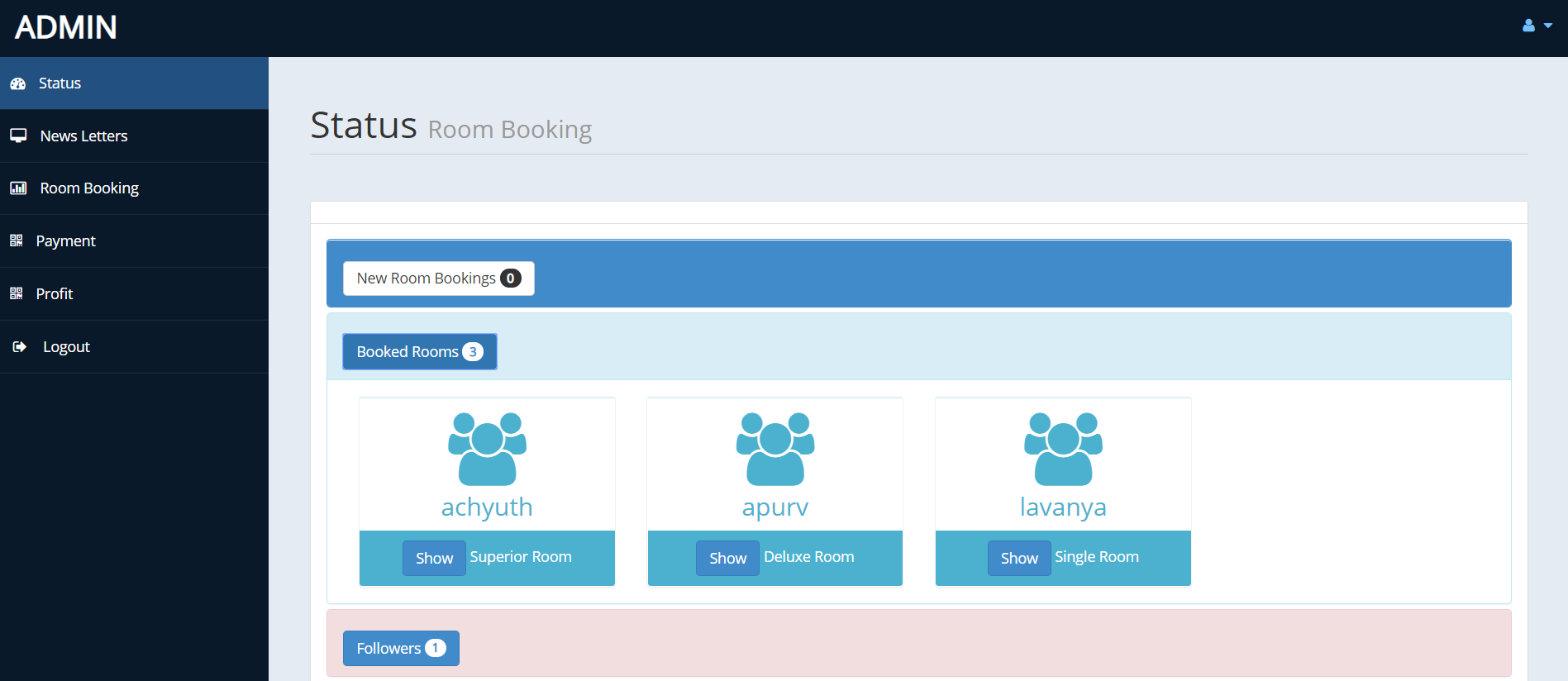
**6.Roombook:** This table contains id, type, bedding, place, cusid. Where id is the primary key.

**4.3 Discussion of Result**

**4.3.1 Admin Login Page:**

**Fig 4.3.1-Login page**

This login page is only for admins or managers of the hotel who all manages the hotel,The admin/manager will be given with unique username and password.Only an admin can add new admins.These admins will approve the reservation of the cutomers by checking the availability.They can also send news letters to customers who signed up for them.

**4.3.2 Admin Homepage:**

**Fig 4.3.2-Admin homepage**

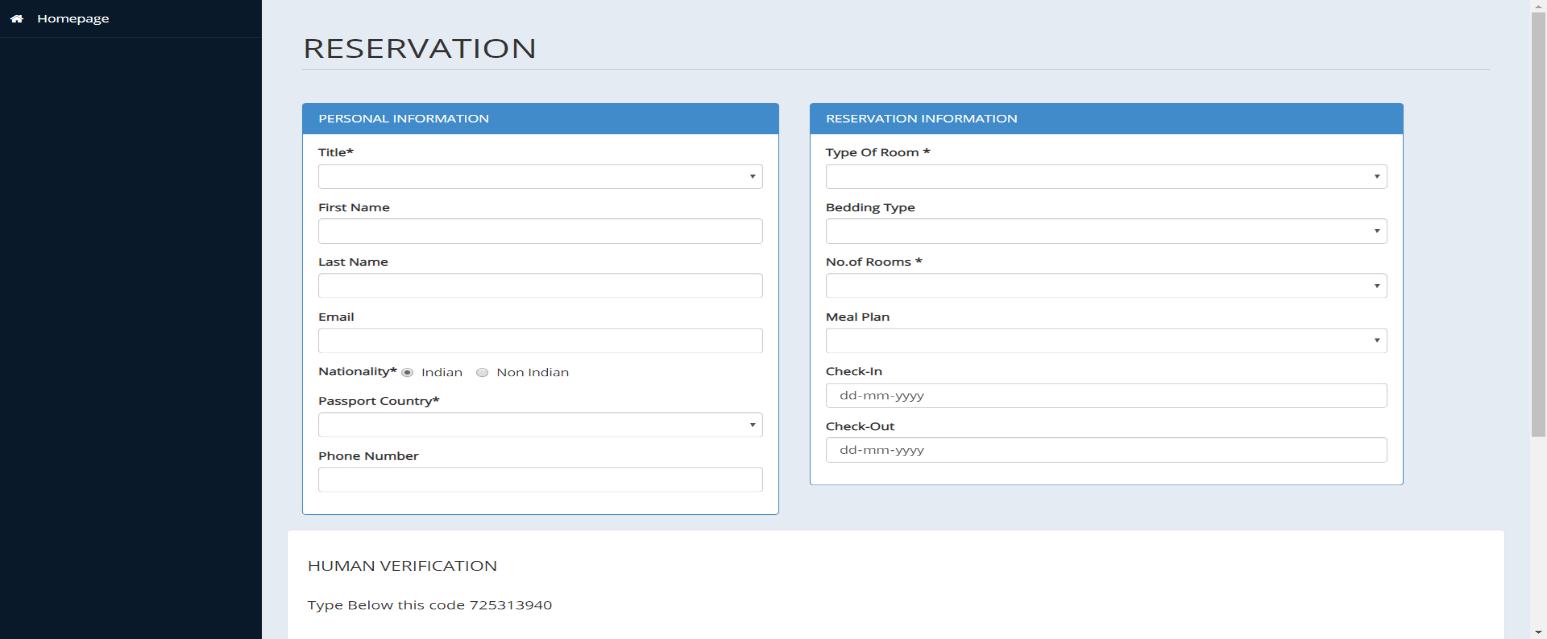
This is admins homepage ,this page will be opened after sucessful login of admins.Here the admins can see new reservation for the hotel room.Admin will approve this based on the room availability.

**4.3.3 Userview of Rooms and Rates:** 

**Fig4.3.2-Room and Rates**

Hotel consists of four different types of room,Superior,Deluxe,Guest,Single Room.The rate of the room differwith each other.Customer can book based on his/her priority.

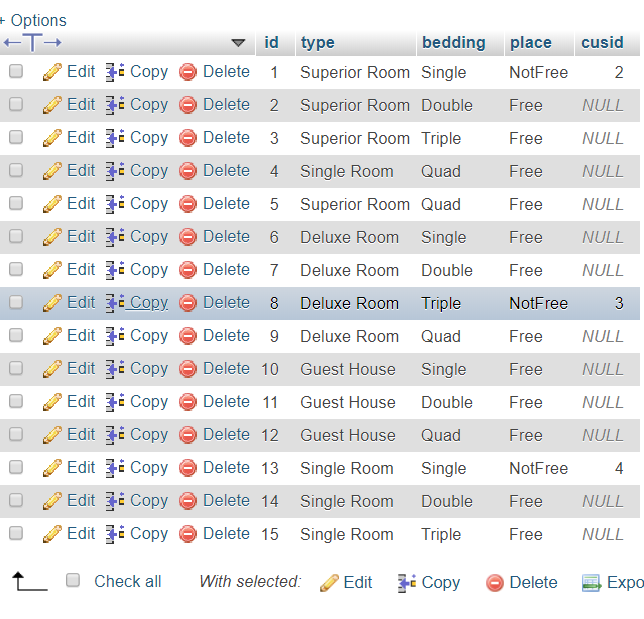
**4.3.4 Resarvation page :**

****

**Fig4.3.4-Reservation**

Figure 1.4 shows Room Reservation page,customer have fill each box compulsorily.Customer will be having sereval choices in Reservation information section.After sending reservation request,customer should wait for manageror admin approval confirmation.

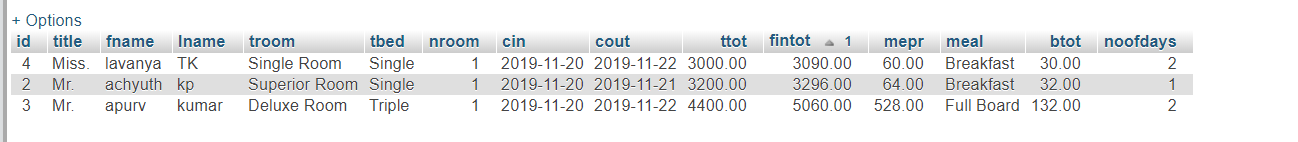
**4.3.5 Rooms availability**



**Fig 4.3.5-Rooms available**

Figure 4.3.5 shows the database content of all the room available for the customer to do reservation.

**4.3.6 payment details in database:**



**Fig 4.3.6-Payment details in database**

This is the database image ,it consists of all the payment details of the customer whose room reservation is Approved/confirmed.

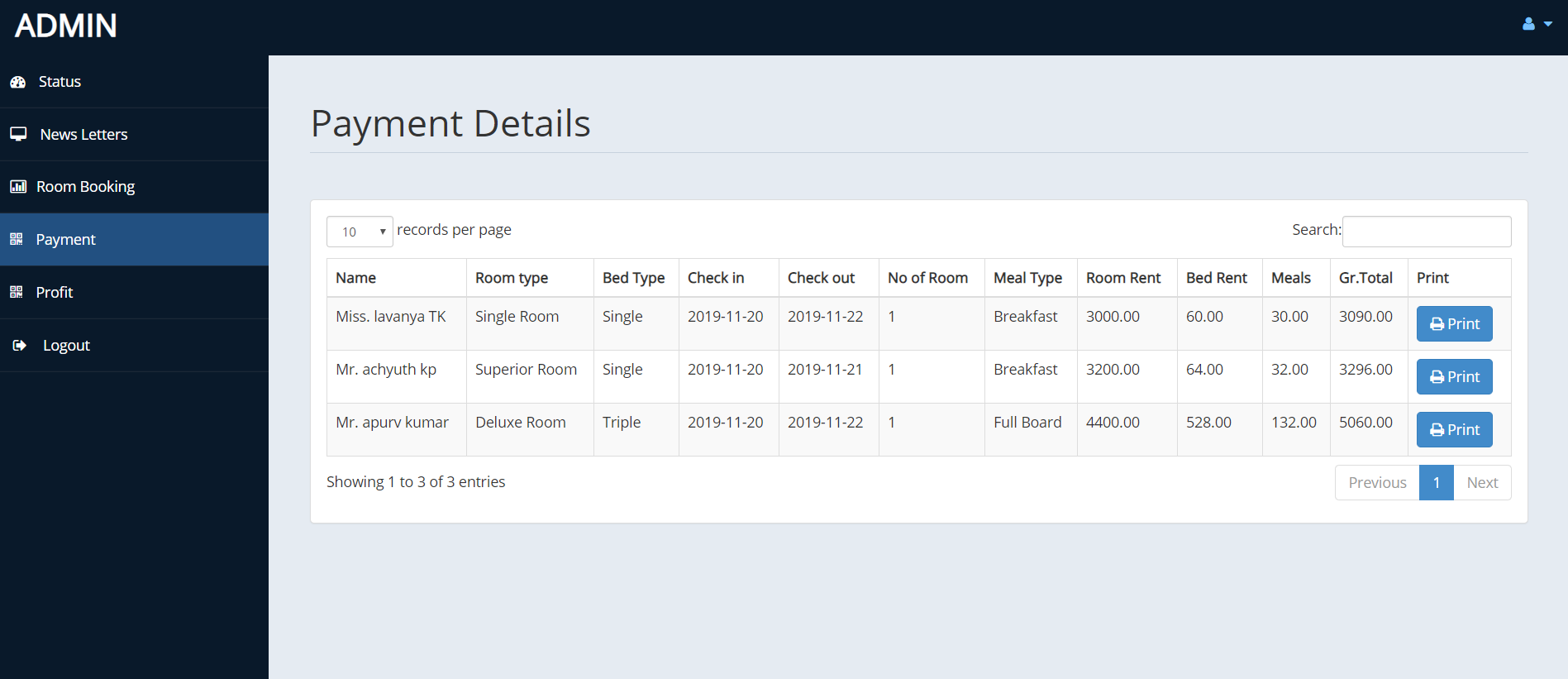
**4.3.7 Reservation details in database:**



**Fig 4.3.7-Reservation details in database**

Figure 4.3.7 consists of the reservation details of the customer and in this it will show the status of the customer room reservation

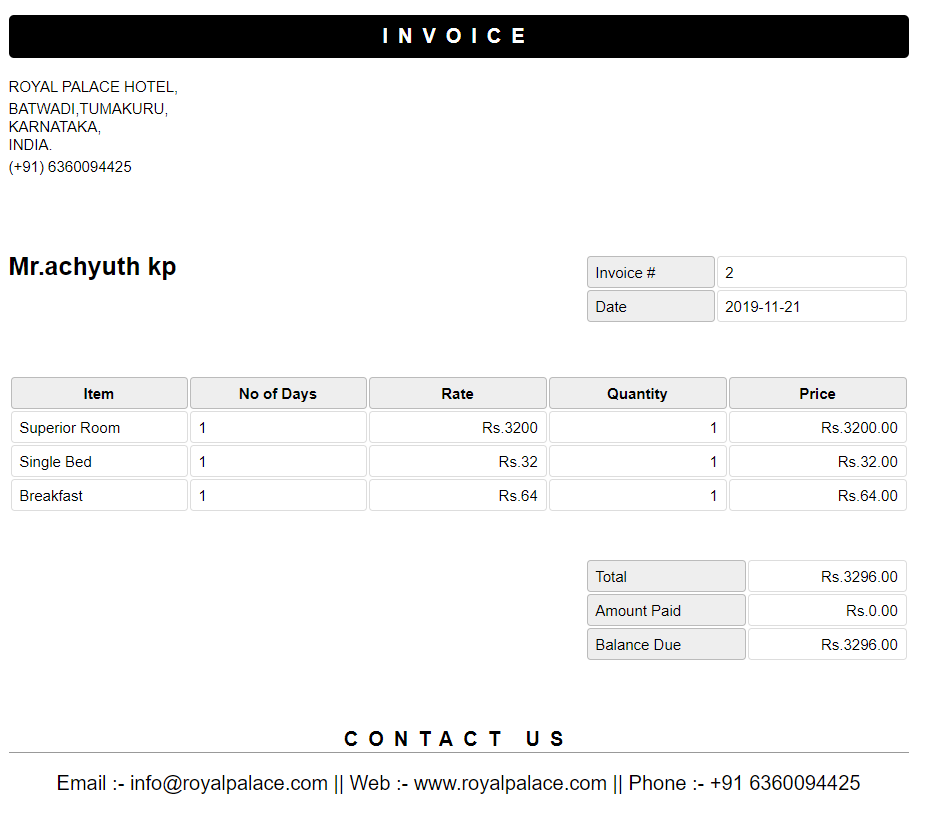
**4.3.8 Payment details of customer in admin page:**



**Fig 4.3.8-Payment details**

This page is payment page which is part of admin,only admins can access this page.Admin can look after the payment details whether the customer paid the amount or yet to be paid.

**4.3.9 Payment Receipt**



**Fig 4.3.9-Invoice**

This is the final page,A payment receipt is a document given to a customer as proof of full or partial pfor the accomidation and any other services he/she availed of at the Hotel.

**Chapter Five**

**Implementation**

**5.1Implementation**

Implementation is the process of having system personal check out and provides new equipment’s into use, train the user to install a new application and construct any files of data needed to use it. There are three types of implementation. Implementation of computer system to replace a manual system. To problem encountered are covering files, training user, creating accurate files and verifying print outs for integrity. Implementation of a new computer system to replace an existing one. This is usually difficult conversion. If not properly planned, there can be many problems. So large computer system many take as long as a year to convert. Implementation of a modified application to replace the existing one using the same computer. This type of conversing is relatively easy to handle, usually there are no major change in the file. Our project is yet to be implemented.

The implementation view of software requirement presents the real world manifestation of processing functions and information structures. This computerized system is specified in a manner that dictates accommodation of certain implementation details.

The implementation environment of the developed system facilitates multiple users to use this system simultaneously. The user interfaces are designed keeping in mind that the users of this system are familiar to using GUI-based systems. Thus, we restricted ourselves to developing a GUI-based system so that it becomes easier for the end user to get acquainted to the developed system.

This system interface is divided into two section

1. Administrator interface.
2. Users interface.

**5.3.1 Administrator Interface**

* Administrator can delete any post.
* Administrator can verified user account.

**5.3.2 User Interface**

1. User can browse all ads without any account.
2. For post an ad needs to create an account
3. User can update/edit their own account.
4. Log in and Log out system.
5. To create a new account user must be needs to verify his email with verification code.
6. If any user forget his/her password he/she can recovery his account with verify his email and create a new password.

This system interface is divided into two section

1. Administrator interface.
2. Users interface.

**5.3.1 Administrator Interface**

* Administrator can delete any post.
* Administrator can verified user account.

**5.3.2 User Interface**

1. User can browse all ads without any account.
2. For post an ad needs to create an account
3. User can update/edit their own account.
4. Log in and Log out system.
5. To create a new account user must be needs to verify his email with verification code.
6. If any user forget his/her password he/she can recovery his account with verify his email and create a new password.

***5.4 Front-end and Back-end used***

*We have used HTML and CSS in the front end. HTML is used to design or create the web pages and CSS is used to style the web pages. For the connection purpose we have been using PHP. i.e. to connect back and front ends*.

***5.4.1 Front****-****end: HTML, CSS***

*HTML is used as the front-end tool to design web pages because:*

* *It is easy enough to write, use and understand*
* *HTML also allows the use of templates, which makes designing a webpage easy*
* *All browsers support HTML*

*CSS is used along with html to design the web pages as it’s relatively easy to learn and produces better and* *cleaner code than applying all those styles directly to your HTML:*

* *Easy to maintain and update*
* *Greater consistency in design and more formatting options*
* *Greater accessibility*

***5.4.2 Back****-****end: PHP and MySQL***

*MySQL is a free-to-use, open-source database that facilitates effective management of databases by connecting them to the software. It is a stable, reliable and powerful solution with advanced features like the following:*

* *MySQL is globally renowned for being the most secure and reliable DBMS used in popular web applications.*
* *MySQL features a distinct storage-engine framework that facilitates system*

*administrators to configure the MySQL database server for a flawless performance.*

* *MySQL tops the list of robust transactional database engines available on the market with features like complete atomic, consistent, isolated, durable transaction support.*

*PHP (Hypertext Pre-Processor) is a server-side web programming language that is widely used for web*

*development. MySQL is used with PHP as the back end tool.*

* *PHP also has powerful output buffering that further increases over the output flow.*
* *PHP is dynamic. PHP works in combination of HTML to display dynamic elements on the page.*
* *PHP can be used with a large number of relational database management systems, runs on all*

*of the most popular web servers and is available for many different operating systems.*

**Chapter six**

**PHP CODE:**

**6.1 Home.php:**

<?php session\_start(); if(!isset($\_SESSION["user"])){ header("location:index.php");}?>

<!DOCTYPE html><html xmlns="http://www.w3.org/1999/xhtml">

<head><meta charset="utf-8" />

<title>Administrator</title> </head>

<body>

<ul class="dropdown-menu dropdown-user">

<li><a href="usersetting.php"><i class="fa fa-user fa-fw"></i> User Profile</a> </li>

<li><a href="settings.php"><i class="fa fa-gear fa-fw"></i> Settings</a> </li>

<li class="divider"></li><li><a href="logout.php"><i class="fa fa-sign-out fa-fw"></i> Logout</a></li></ul>

<li><a class="active-menu" href="home.php"><i class="fa fa-dashboard"></i> Status</a></li> <li>

<a href="messages.php"><i class="fa fa-desktop"></i> News Letters</a></li>

<li> <a href="roombook.php"><i class="fa fa-bar-chart-o"></i> Room Booking</a>

</li> <li> <a href="payment.php"><i class="fa fa-qrcode"></i> Payment</a>

</li><li><a href="profit.php"><i class="fa fa-qrcode"></i> Profit</a></li>

<li><a href="logout.php"><i class="fa fa-sign-out fa-fw"></i> Logout</a></li></ul>

</div>

<!-- /. ROW -->

<?php include ('db.php');

$sql = "select \* from roombook";

$re = mysqli\_query($con,$sql);

$c =0;

while($row=mysqli\_fetch\_array($re) )

{

$new = $row['stat'];

$cin = $row['cin'];

$id = $row['id'];

if($new=="Not Conform")

{ $c = $c + 1;

}}?>

<div class="row"> <div class="col-md-12"> <div class="panel panel-default">

<div class="panel-heading">

</div> <div class="panel-body"> <div class="panel-group" id="accordion">

<div class="panel panel-primary"> <div class="panel-heading">

<h4 class="panel-title">

<a data-toggle="collapse" data-parent="#accordion" href="#collapseTwo">

<button class="btn btn-default" type="button"> New Room Bookings <span class="badge"><?php echo $c ; ?></span></button></a> </h4> </div>

<div id="collapseTwo" class="panel-collapse in" style="height: auto;">

<div class="panel-body"> <div class="panel panel-default">

<div class="panel-body"><div class="table-responsive">

<table class="table"> <thead> <tr> <th>#</th> <th>Name</th>

<th>Email</th> <th>Country</th> <th>Room</th>

<th>Bedding</th> <th>Meal</th> <th>Check In</th> <th>Check Out</th>

<th>Status</th> <th>More</th> </tr></thead>

<tbody>

<?php $tsql = "select \* from roombook";

$tre = mysqli\_query($con,$tsql);

while($trow=mysqli\_fetch\_array($tre) ) {

$co =$trow['stat'];

if($co=="Not Conform"){

echo"<tr><th>".$trow['id']."</th>

<th>".$trow['FName']." ".$trow['LName']."</th>

<th>".$trow['Email']."</th> <th>".$trow['Country']."</th> <th>".$trow['TRoom']."</th> <th>".$trow['Bed']."</th> <th>".$trow['Meal']."</th> <th>".$trow['cin']."</th>

<th>".$trow['cout']."</th> <th>".$trow['stat']."</th>

<th><a href='roombook.php?rid=".$trow['id']." ' class='btn btn-primary'>Action</a></th>

</tr>"; } } ?>

</tbody> </table> </div> </div> </div> <!-- End Basic Table --> </div>

</div> </div>

<?php

$rsql = "SELECT \* FROM `roombook`"; $rre = mysqli\_query($con,$rsql);

$r =0; while($row=mysqli\_fetch\_array($rre) ) {

$br = $row['stat'];

if($br=="Conform")

{

$r = $r + 1; } } ?

<div class="panel panel-info"> <div class="panel-heading"> <h4 class="panel-title">

<a data-toggle="collapse" data-parent="#accordion" href="#collapseOne" class="collapsed"> <button class="btn btn-primary" type="button">

Booked Rooms <span class="badge"><?php echo $r ; ?></span></button>

</a></h4>

</div>

<div id="collapseOne" class="panel-collapse collapse" style="height: 0px;">

<div class="panel-body">

</body>

</html>

**6.2 Logout.php:**

<?php

session\_start();

unset($\_SESSION["user"]);

header("location:index.php"); ?>

**6.3 Room.php:**

<?php session\_start();

if(!isset($\_SESSION["user"]))

{ header("location:index.php"); } ?>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>ROYALPALACE HOTEL</title>

<a class="navbar-brand" href="home.php">MAIN MENU </a> </div>

<ul class="nav navbar-top-links navbar-right">

<li class="dropdown">

<a class="dropdown-toggle" data-toggle="dropdown" href="#" aria-expanded="false">

<i class="fa fa-user fa-fw"></i> <i class="fa fa-caret-down"></i> </a>

<ul class="dropdown-menu dropdown-user">

<li><a href="usersetting.php"><i class="fa fa-user fa-fw"></i> User Profile</a> </li>

<li><a href="settings.php"><i class="fa fa-gear fa-fw"></i> Settings</a> </li>

<li class="divider"></li> <li><a href="logout.php"><i class="fa fa-sign-out fa-fw"></i> Logout</a></li></ul>

<?php

include('db.php');

if(isset($\_POST['add'])) {

$room = $\_POST['troom'];

$bed = $\_POST['bed'];

$place = 'Free';

$check="SELECT \* FROM room WHERE type = '$room' AND bedding = '$bed'";

$rs = mysqli\_query($con,$check);

$data = mysqli\_fetch\_array($rs, MYSQLI\_NUM);

if($data[0] > 1) {  
echo "<script type='text/javascript'> alert('Room Already in Exists')</script>";

}

else

{

$sql ="INSERT INTO `room`( `type`, `bedding`,`place`) VALUES ('$room','$bed','$place')" ;

if(mysqli\_query($con,$sql))

{

echo '<script>alert("New Room Added") </script>' ; }else {

echo '<script>alert("Sorry ! Check The System") </script>' ; } } } ?>

</div> </div> </div>

<div class="row">

<div class="col-md-6 col-sm-6">

<div class="panel panel-primary">

<div class="panel-heading">

ROOMS INFORMATION

</div>

<div class="panel-body"> <!-- Advanced Tables -->

<div class="panel panel-default">

<?php

$sql = "select \* from room limit 0,10";

$re = mysqli\_query($con,$sql) ?>

<?php  
while($row= mysqli\_fetch\_array($re)) {

$id = $row['id'];

if($id % 2 == 0) {

echo "<tr class=odd gradeX>

<td>".$row['id']."</td>

<td>".$row['type']."</td> <th>".$row['bedding']."</th> </tr>"; }

else {

echo"<tr class=even gradeC>

<td>".$row['id']."</td>

<td>".$row['type']."</td> <th>".$row['bedding']."</th> </tr>";

} } } ?>

</tbody>

</table> </div> </div> </div> <!--End Advanced Tables --> </div></div></div></div>

</body>

</html>

**6.4 Payment.php:**

<?php

session\_start();

if(!isset($\_SESSION["user"]))

{

header("location:index.php");

} ?>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>ROYALPALACE HOTEL</title>

<a class="navbar-brand" href="home.php"><?php echo $\_SESSION["user"]; ?> </a>

</div> <ul class="nav navbar-top-links navbar-right">

<li class="dropdown">

<a class="dropdown-toggle" data-toggle="dropdown" href="#" aria-expanded="false">

<i class="fa fa-user fa-fw"></i> <i class="fa fa-caret-down"></i> </a>

<ul class="dropdown-menu dropdown-user">

<li><a href="usersetting.php"><i class="fa fa-user fa-fw"></i> User Profile</a> </li>

<li><a href="settings.php"><i class="fa fa-gear fa-fw"></i> Settings</a> </li>

<li class="divider"></li>

<li><a href="logout.php"><i class="fa fa-sign-out fa-fw"></i> Logout</a></li></ul>

<?php

include ('db.php');

$sql="select \* from payment";

$re = mysqli\_query($con,$sql);

while($row = mysqli\_fetch\_array($re)) {

$id = $row['id'];

if($id % 2 ==1 )

echo"<tr class='gradeC'>

<td>".$row['title']." ".$row['fname']." ".$row['lname']."</td> <td>".$row['troom']."</td> <td>".$row['tbed']."</td> <td>".$row['cin']."</td> <td>".$row['cout']."</td>

<td>".$row['nroom']."</td> <td>".$row['meal']."</td>

<td>".$row['ttot']."</td> <td>".$row['mepr']."</td> <td>".$row['btot']."</td>

<td>".$row['fintot']."</td> <td><a href=print.php?pid=".$id ." <button class='btn btn-primary'> <i class='fa fa-print' ></i> Print</button></td> </tr>"; }

else

{

echo"<tr class='gradeU'>

<td>".$row['title']." ".$row['fname']." ".$row['lname']."</td>

<td>".$row['troom']."</td> <td>".$row['tbed']."</td> <td>".$row['cin']."</td>

<td>".$row['cout']."</td> <td>".$row['nroom']."</td> <td>".$row['meal']."</td>

<td>".$row['ttot']."</td> <td>".$row['mepr']."</td> <td>".$row['btot']."</td>

<td>".$row['fintot']."</td>

<td><a href=print.php?pid=".$id ." <button class='btn btn-primary'> <i class='fa fa-print' ></i> Print</button></td>

</tr>"; } } } ?>

</tbody> </table> </div> </div>

</div> <!--End Advanced Tables --> </div> </div>

</body>

</html>

**6.5 Reservation.php:**

<?php

include('db.php')

?>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>RESERVATION ROYALPALACE HOTEL</title>

<a href="../index.php"><i class="fa fa-home"></i> Homepage</a>

</li> </ul> </div> </nav>

<?php

foreach($countries as $key => $value):

echo '<option value="'.$value.'">'.$value.'</option>'; //close your tags!! endforeach; ?>

</select> </div>

<div class="form-group"> <label>Phone Number</label>

<input name="phone" type ="text" class="form-control" required>

</div> </div> </div> </div>

<div class="row">

<div class="col-md-6 col-sm-6">

<div class="panel panel-primary">

<div class="panel-heading">

RESERVATION INFORMATION

</div>

<div class="panel-body">

<div class="form-group"> <label>Type Of Room \*</label>

<select name="troom" class="form-control" required>

<option value selected ></option> <option value="Superior Room">SUPERIOR ROOM</option> <option value="Deluxe Room">DELUXE ROOM</option>

<option value="Guest House">GUEST HOUSE</option>

<option value="Single Room">SINGLE ROOM</option> </select>

</div> <div class="form-group">

<label>Bedding Type</label> <select name="bed" class="form-control" required>

<option value selected ></option> <option value="Single">Single</option>

<option value="Double">Double</option>

<option value="Triple">Triple</option> <option value="Quad">Quad</option>

<option value="None">None</option>

</select> </div>

<div class="form-group"> <label>No.of Rooms \*</label>

<select name="nroom" class="form-control" required>

<option value selected ></option> <option value="1">1</option> </select> </div> <div class="form-group"> <label>Meal Plan</label>

<select name="meal" class="form-control"required>

<option value selected ></option>

<option value="Room only">Room only</option>

<option value="Breakfast">Breakfast</option>

<option value="Half Board">Half Board</option>

<option value="Full Board">Full Board</option>

</select> </div>

<div class="form-group">

<label>Check-In</label>

<input name="cin" type ="date" class="form-control">

</div>

<div class="form-group">

<label>Check-Out</label>

<input name="cout" type ="date" class="form-control">

</div> </div> </div> </div>

<div class="col-md-12 col-sm-12">

<div class="well">

<h4>HUMAN VERIFICATION</h4>

<p>Type Below this code <?php $Random\_code=rand(); echo$Random\_code; ?> </p><br />

<p>Enter the random code<br /></p>

<input type="text" name="code1" title="random code" />

<input type="hidden" name="code" value="<?php echo $Random\_code; ?>" />

<input type="submit" name="submit" class="btn btn-primary">

<?php

if(isset($\_POST['submit']))

{

$code1=$\_POST['code1'];

$code=$\_POST['code'];

if($code1!="$code")

{

$msg="Invalide code"; }

else { $con=mysqli\_connect("localhost","root","","hotel");

$check="SELECT \* FROM roombook WHERE email = '$\_POST[email]'";

$rs = mysqli\_query($con,$check);

$data = mysqli\_fetch\_array($rs, MYSQLI\_NUM);

if($data[0] > 1) {

echo "<script type='text/javascript'> alert('User Already in Exists')</script>"; }

else

{

$new ="Not Conform";

$newUser="INSERT INTO `roombook`(`Title`, `FName`, `LName`, `Email`, `National`, `Country`, `Phone`, `TRoom`, `Bed`, `NRoom`, `Meal`, `cin`, `cout`,`stat`,`nodays`) VALUES ('$\_POST[title]','$\_POST[fname]','$\_POST[lname]','$\_POST[email]','$\_POST[nation]','$\_POST[country]','$\_POST[phone]','$\_POST[troom]','$\_POST[bed]','$\_POST[nroom]','$\_POST[meal]','$\_POST[cin]','$\_POST[cout]','$new',datediff('$\_POST[cout]','$\_POST[cin]'))";

if (mysqli\_query($con,$newUser)) {

echo "<script type='text/javascript'> alert('Your Booking application has been sent')</script>";

}

else

{

echo "<script type='text/javascript'> alert('Error adding user in database')</script>"; } }

$msg="Your code is correct"; } } } ?>

</form>

</div> </div> </div> </div> </div>

</body>

</html>

**Chapter Seven**

**Conclusion**

**7.1 Conclusions**

This project has been a rewarding experience in more than one way. The entire project work has enlightened us in the following areas.

1. We have gained an insight into the working of the HOTEL. This represents a typical real world situation.
   1. Our understanding of database design has been strengthened this is because in order to generate the final reports of database designing has to be properly followed.
   2. Scheduling a project and adhering to that schedule creates a strong sense of time management.
   3. Sense of teamwork has developed and confidence of handling real life project has increased to a great extent.
   4. Initially, there were problem with the validation but with discussions, we were to implement validations.

**7.2 Limitations of the system**

1. Online payment is not available at this version.
2. Data delete & edit system is not available for all section.
3. User account not verified by Mobile SMS not available in this system.
4. Loss of data due to mismanagement.

**References**

* Deepak Thomas ͞”Beginning PHP 4 Databases”, Wrox Press Ltd. Paperback-17, October, 2002.70-130 pp.
* Matt Doyle, “Beginning PHP 5.3, 2ndedition”, October 2009. 150-270 pp.
* Luke Welling, Laura Thomson. Sams PHP͞ and MySQL Web Development, 2nd edition, Paperback- 20 February, 2003. 105-209 pp.
* W. Jason Gilmore “Beginning PHP 5 and MySQL 5 from Novice to Professional SECOND EDITION”, Jul 9, 2008.100-150 pp.
* Abraham Silberschatz, Henry F. Korth and S. Sudarshan “Sixth Edition Database System Conceptsreleased”, January 28, 2010. 206-253 pp.
* Server-Side [Scriptinghttp://php.net/manual/en/index.php,](http://php.net/manual/en/index.php) Last accessed on 12/15/2017at 2:33pm
* HTML & CSS <https://ww.w3schools.com/>
* Bootstrap <http://getbootstrap.com/>
* Stackoverflow http://stackoverflow.com/