

# DBMS CASE STUDY- COMPANY DEALING

**GROUP NUMBER 7**

**05-11-21**

**B4**

Teammates	Registration Number
Iti Manasa	121910304027
S Naman	121910304061
Sri ram	121910304045
D.B.M Madhu	121910304060
Navya Sree	121910304032

Topic	Page Number
Abstract	02
Requirement analysis	03
Module description	03
Conceptual database	04
Logical database	06
Sample data	13
Execution – sql queries	23
Scheme refinement -Normalization	30
Finding Candidate Key	32
conclusion	34
References	28

## ABSTRACT

A relational database is to be designed for a medium sized Company dealing with industrial applications of computers. The Company delivers various products to its customers ranging from a single application program through to complete installation of hardware with customized software. The Company employs various experts, consultants and supporting staff. All personnel are employed on long term basis, i.e . there are no short-term or temporary staff. Although the Company is somehow structured for administrative purposes (that is, it is divided into departments headed by department managers) all projects are carried out in an inter-disciplinary way. For each project a project team is selected, grouping employees from different departments, and a Project Manager (also an employee of the Company) is appointed who is entirely and exclusively responsible for the control of the project, quite independently of the Company's hierarchy. The following is a brief statement of some facts and policies adopted by the Company.

- Each employee works in some departments.
- An employee may possess a number of skills.
- Every manager is an employee.
- A department may participate in none/one/many projects.
- At least one department participates in a project.
- An employee may be engaged in none/one/many projects.
- Project teams consist of at least one member

To implement this sample case study, some assumptions have been made.

- 1) The company has six departments. They are Software, Testing, Finance, Customer Support, IOS development, Android Develop department.
- 2) The company is currently working on three projects. They are “Developing an Android app for a hotel”, “Developing website for a retail store”, “Developing an IOS app for a school”.
- 3) There are total 23 employees working in the company.

## REQUIREMENT ANALYSIS

### **Existing system :**

The existing system starts with companies when they start to recruit people in their companies. Each employee works in some department and that employee may possess a number of skills. Companies give projects to employees to work on them and to finish them. An employee may be engaged in one or many projects according to his/her talent or even the complexity of the project.

### **Problems in the existing system :**

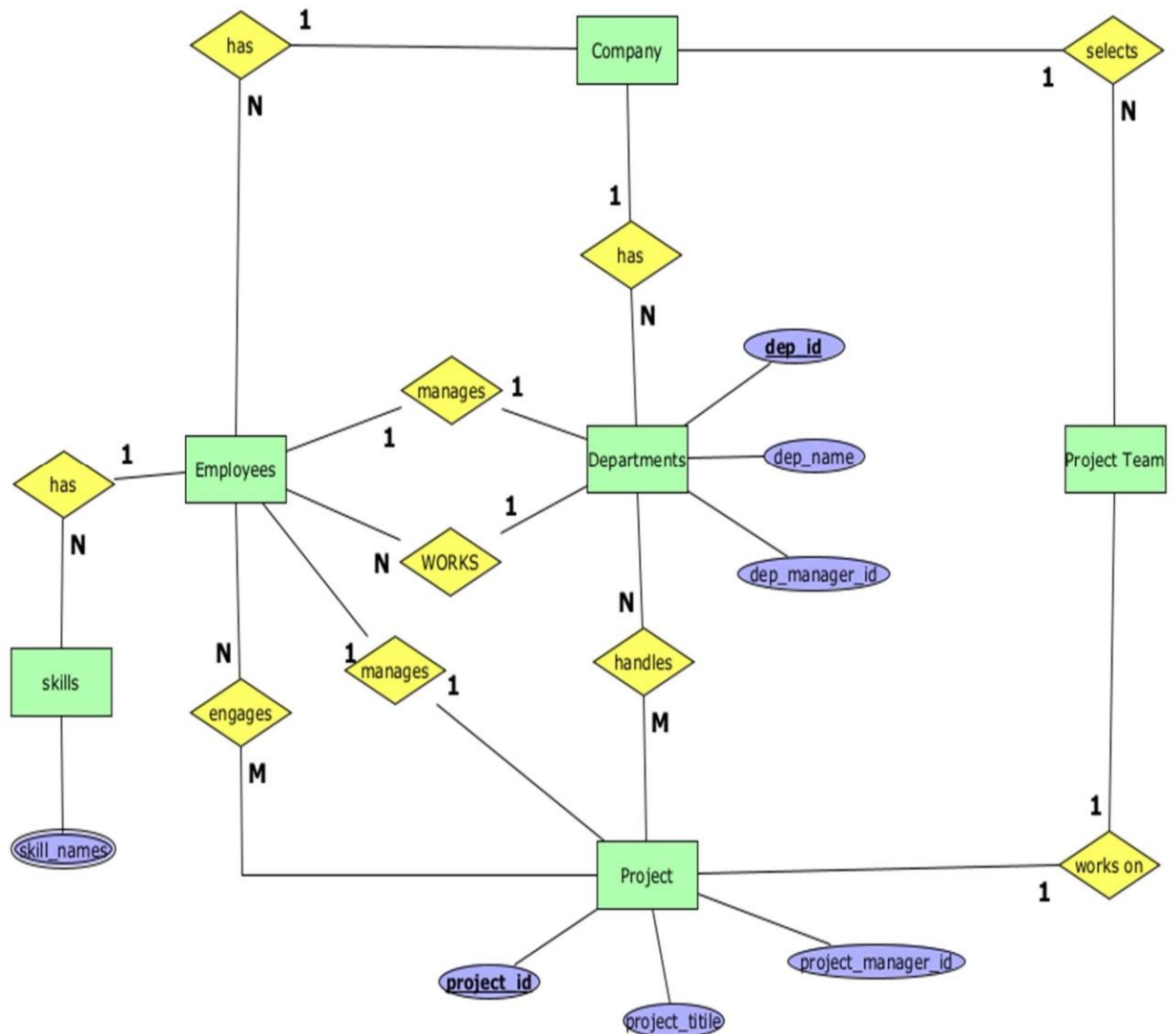
Storing and accessing the data in the form of Excel sheets and account books is a tedious work. It requires a lot of laborious work. It may often yield undesired results. Maintaining these records as piles may turn out to be a costlier task than any other of the colleges and institutions.

## MODULE DESCRIPTION

1. Employee – he works in the company and becomes a part of a group and complete the project.
2. Department – Every employee belongs to some or the other department, so that it would be easy to assign task based on their skill.
3. Company – The company recruits people and makes them its employees, it has many employees and departments.
4. Project – A project is given to a group of employees, and they should be working on it and should complete it on time.
5. Project Team – A team will be selected in a such a way that they should be able to complete the given task.

# CONCEPTUAL DATABASE DESIGN

## 1 ER MODEL :



## 2 ENTITIES AND ATTRIBUTES LIST:

Entity	Attributes
Department	<u>Dep_ID</u> Dep_Name
Employee	<u>Emp_ID</u> Emp_Name Salary Dep_ID
Department_Manager	<u>Dep_ID</u> Dep_manager_ID
Project	<u>Project_ID</u> Project_Title Project_manager_ID
Employee_Skills	<u>Emp_ID</u> Skill_Names
Works_On	<u>Emp_ID</u> <u>Project_ID</u>

## 3 RELATIONS:

1. Company has many employees.
2. Company has many departments.
3. Many departments handle many projects.
4. Department has only one manager.
5. Project has only one manager.
6. Each employee has many skills.
7. Each employee is a part of only one department.
8. Each employee may work in none/one/many projects.

# LOGICAL DATABASE DESIGN

## 1 RELATIONAL SCHEMA:

DEPARTMENT(DEP\_ID, DEP\_NAME);

EMPLOYEE(EMP\_ID, EMP\_NAME);

DEP\_MANAGER(DEP\_MANAGER\_ID, DEP\_ID);

PROJECT(PROJECT\_ID, PROJECT\_TITLE, PRJ\_MANAGER\_ID);

EMPLOYEE\_SKILLS(SKILL\_NAMES, EMP\_ID);

WORKS\_ON(PROJECT\_ID, EMP\_ID);

## 2 TABLES CREATION:

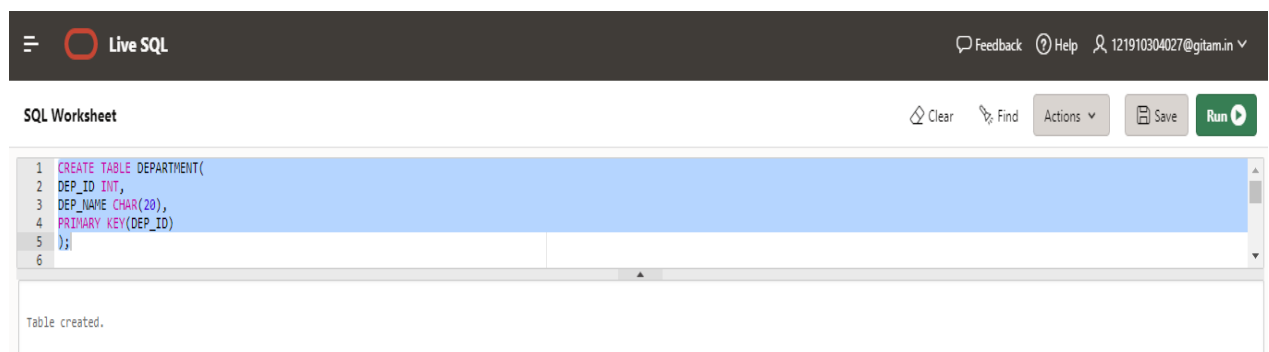
CREATE TABLE DEPARTMENT(

DEP\_ID INT,

DEP\_NAME CHAR(20),

PRIMARY KEY(DEP\_ID)

);





The screenshot shows a web-based SQL editor interface. At the top, there's a dark header with a hamburger menu, a "Live SQL" logo, and links for Feedback, Help, and a user profile. Below the header, the main area is titled "SQL Worksheet". It contains a text editor with the following SQL code: 

```
1 CREATE TABLE DEPARTMENT(  
2 DEP_ID INT,  
3 DEP_NAME CHAR(20),  
4 PRIMARY KEY(DEP_ID)  
5 );  
6
```

 To the right of the editor are buttons for "Clear", "Find", "Actions", "Save", and a green "Run" button. Below the editor, a status message reads "Table created."

## Desc DEPARTMENT;



Live SQL
Feedback
Help
121910304027@gitam.in

SQL Worksheet
Clear
Find
Actions
Save
Run

```

1 CREATE TABLE DEPARTMENT(
2   DEP_ID INT,
3   DEP_NAME CHAR(20),
4   PRIMARY KEY(DEP_ID)
5 );
6
7 Desc DEPARTMENT;

```

Column	Null?	Type
DEP_ID	NOT NULL	NUMBER
DEP_NAME	-	CHAR(20)

Download CSV  
2 rows selected.

Column_name	Data Type	Contrain_t_type
DEPT_ID	INT	PRIMARY KEY
DEPT_NAME	CHAR(20)	NOT NULL

```

CREATE TABLE EMPLOYEE( EMP_ID INT,

EMP_NAME CHAR(30),

SALARY FLOAT,
DEP_ID INT,
PRIMARY KEY(EMP_ID),
FOREIGN KEY (DEP_ID) REFERENCES DEPARTMENT (DEP_ID)
);

```

```

18
19 CREATE TABLE EMPLOYEE(
20   EMP_ID INT,
21   EMP_NAME CHAR(30),
22   SALARY FLOAT,
23   DEP_ID INT,
24   PRIMARY KEY(EMP_ID),
25   FOREIGN KEY (DEP_ID) REFERENCES DEPARTMENT (DEP_ID)
26 );
27

```

Table created.



```
desc EMPLOYEE;
```

27	desc EMPLOYEE;
28	

Column	Null?	Type
EMP_ID	NOT NULL	NUMBER
EMP_NAME	-	CHAR(30)
SALARY	-	FLOAT(126)
DEP_ID	-	NUMBER

Column_name	Data Type	Constraint_type
EMP_ID	INT	PRIMARY KEY
NAME	CHAR(30)	NOT NULL
SALARY	INT	NOT NULL
DEPT_ID	INT	FOREIGN KEY (references from DEPARTMENT table)

```
CREATE TABLE DEP_MANAGER(
DEP_MANAGER_ID INT,
DEP_ID INT,
PRIMARY KEY(DEP_MANAGER_ID),
FOREIGN KEY (DEP_MANAGER_ID) REFERENCES EMPLOYEE (EMP_ID)
);
```

```

61 CREATE TABLE DEP_MANAGER(
62 DEP_MANAGER_ID INT,
63 DEP_ID INT,
64 PRIMARY KEY(DEP_MANAGER_ID),
65 FOREIGN KEY (DEP_MANAGER_ID) REFERENCES EMPLOYEE (EMP_ID)
66 );
67

```

Table created.

Decs DEP\_MANAGER;

```

67
68 desc DEP_MANAGER;
69
70

```

TABLE DEP\_MANAGER

Column	Null?	Type
DEP_MANAGER_ID	NOT NULL	NUMBER
DEP_ID	-	NUMBER

[Download CSV](#)

2 rows selected.

Col Name	Data type	Contrain_type
DEP_MANAGER_id	int	Primary Key, FOREIGN KEY (references from employee table)
Dep_ID	int	Primary key

```
CREATE TABLE PROJECT( PROJECT_ID INT,
PROJECT_TITLE CHAR(50), PRJ_MANAGER_ID INT, PRIMARY KEY(PROJECT_ID),
FOREIGN KEY (PRJ_MANAGER_ID) REFERENCES EMPLOYEE (EMP_ID)
);
```

```
81 CREATE TABLE PROJECT(
82 PROJECT_ID INT,
83 PROJECT_TITLE CHAR(50),
84 PRJ_MANAGER_ID INT,
85 PRIMARY KEY(PROJECT_ID),
86 FOREIGN KEY (PRJ_MANAGER_ID) REFERENCES EMPLOYEE (EMP_ID)
87 );
88
```

Table created.

Desc PROJECT;

```
88 Desc PROJECT;
89
90
```

TABLE PROJECT

Column	Null?	Type
PROJECT_ID	NOT NULL	NUMBER
PROJECT_TITLE	-	CHAR(50)
PRJ_MANAGER_ID	-	NUMBER

[Download CSV](#)  
3 rows selected.

Column_name	Data Type	Contrain_type
PRJ_ID	INT	PRIMARY KEY
PRJ_TITLE	CHAR(50)	NOT NULL
PRJ_MANAGER_ID	INT	FOREIGN KEY (references from EMPLOYEE table)

```
CREATE TABLE EMPLOYEE_SKILLS(
SKILL_NAMES CHAR(50),
EMP_ID INT,
PRIMARY KEY(EMP_ID),
FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID)
);
```

```
98 CREATE TABLE EMPLOYEE_SKILLS(
99 SKILL_NAMES CHAR(50),
100 EMP_ID INT,
101 PRIMARY KEY(EMP_ID),
102 FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID)
103 );
104
```

Table created.

Desc EMPLOYEE\_SKILLS;

```
104 Desc EMPLOYEE_SKILLS;
```

TABLE EMPLOYEE\_SKILLS

Column	Null?	Type
SKILL_NAMES	-	CHAR(50)
EMP_ID	NOT NULL	NUMBER

[Download CSV](#)  
2 rows selected.

Column_name	Data Type	Contraint_type
SKILL_NAME	CHAR(50)	NOT NULL
EMP_ID	INT	Primary Key, FOREIGN KEY (references from EMPLOYEE table)

```
CREATE TABLE WORKS_ON (

PROJECT_ID INT,
EMP_ID INT,
PRIMARY KEY(PROJECT_ID, EMP_ID),
FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID)
);
```

```
136 CREATE TABLE WORKS_ON (
137 PROJECT_ID INT,
138 EMP_ID INT,
139 PRIMARY KEY(PROJECT_ID, EMP_ID),
140 FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID)
141 );
142
```

Table created.

Desc WORKS\_ON;

```
142 Desc WORKS_ON;
```

TABLE WORKS\_ON

Column	Null?	Type
PROJECT_ID	NOT NULL	NUMBER
EMP_ID	NOT NULL	NUMBER

[Download CSV](#)

2 rows selected.

Column_name	Data Type	Constraint_type
EMP_ID	INT	Primary Key, FOREIGN KEY (references from EMPLOYEE table)
PRJ_ID	INT	Primary Key, FOREIGN KEY (references from PROJECT table)

## SAMPLE DATA

```
INSERT INTO DEPARTMENT VALUES(4050, 'SOFTWARE_DEVELOPMENT');
```

```
INSERT INTO DEPARTMENT VALUES(4065, 'TESTING_TEAM');
```

```
INSERT INTO DEPARTMENT VALUES(4085, 'FINANCE_DEPARTMENT');
```

```
INSERT INTO DEPARTMENT VALUES(6523, 'CUSTOMER_SUPPORT');
```

```
INSERT INTO DEPARTMENT VALUES(7361, 'IOS_Development');
```

```
INSERT INTO DEPARTMENT VALUES(4362, 'ANDROID_DEVELOPMENT');
```

```
SELECT * FROM DEPARTMENT;
```

```

9  INSERT INTO DEPARTMENT VALUES(4050, 'SOFTWARE_DEVELOPMENT');
10 INSERT INTO DEPARTMENT VALUES(4065, 'TESTING_TEAM');
11 INSERT INTO DEPARTMENT VALUES(4085, 'FINANCE_DEPARTMENT');
12 INSERT INTO DEPARTMENT VALUES(6523, 'CUSTOMER_SUPPORT');
13 INSERT INTO DEPARTMENT VALUES(7361, 'IOS_Development');
14 INSERT INTO DEPARTMENT VALUES(4362, 'ANDROID_DEVELOPMENT');
15
16 SELECT * FROM DEPARTMENT;
--

```

DEP_ID	DEP_NAME
4050	SOFTWARE_DEVELOPMENT
4085	FINANCE_DEPARTMENT
7361	IOS_Development
6523	CUSTOMER_SUPPORT
4065	TESTING_TEAM
4362	ANDROID_DEVELOPMENT

[Download CSV](#)  
 6 rows selected.

-- Testing Department

INSERT INTO EMPLOYEE VALUES(84536, 'TANMAY', 50000, 4065);

INSERT INTO EMPLOYEE VALUES(84706, 'VISHAL', 60000, 4065);

INSERT INTO EMPLOYEE VALUES(65034, 'BEN', '35000', 4065);

-- Software Dev Department

INSERT INTO EMPLOYEE VALUES(75346, 'CHIRAG', 55000, 4050);

INSERT INTO EMPLOYEE VALUES(25634, 'JOHN', 65000, 4050);

INSERT INTO EMPLOYEE VALUES(75446, 'MARK', 70000, 4050);

INSERT INTO EMPLOYEE VALUES(55362, 'JAMES', 85000, 4050);

-- Customer Support Department

INSERT INTO EMPLOYEE VALUES(44346, 'RISHAB', 23000, 6523);

INSERT INTO EMPLOYEE VALUES(25346, 'TARUN', 30000, 6523);

INSERT INTO EMPLOYEE VALUES(42316, 'RAVI', 40000, 6523);

-- Finance Department

INSERT INTO EMPLOYEE VALUES(42346, 'RITESH', 55000, 4085);

INSERT INTO EMPLOYEE VALUES(33346, 'TARUN', 55000, 4085); INSERT INTO

EMPLOYEE VALUES(96452, 'RITHIK', 30000, 4085);

INSERT INTO EMPLOYEE VALUES(65321, 'TINA', 65300, 4085);

-- Android Dev Department

INSERT INTO EMPLOYEE VALUES(32645, 'TANISH', 29000, 4362);

INSERT INTO EMPLOYEE VALUES(30270, 'VIVEK', 43000, 4362);

```
INSERT INTO EMPLOYEE VALUES(13640, 'JESSICA', 65000, 4362);
```

```
INSERT INTO EMPLOYEE VALUES(52600, 'VIJAY', 35000, 4362);
```

```
INSERT INTO EMPLOYEE VALUES(45209, 'NEHA', 45000, 4362);
```

```
-- IOS Dev Department
```

```
INSERT INTO EMPLOYEE VALUES(65320, 'VERONICA', 100000, 7361);
```

```
INSERT INTO EMPLOYEE VALUES(15962, 'TIM', 55000, 7361);
```

```
INSERT INTO EMPLOYEE VALUES(63705, 'KRITI', 25000, 7361);
```

```
INSERT INTO EMPLOYEE VALUES(29043, 'EMILY', 65000, 7361);
```

```
SELECT * FROM EMPLOYEE;
```

EMP_ID	EMP_NAME	SALARY	DEP_ID
84706	VISHAL	60000	4065
65034	BEN	35000	4065
25634	JOHN	65000	4050
55362	JAMES	85000	4050
44346	RISHAB	23000	6523
25346	TARUN	30000	6523
42316	RAVI	40000	6523
42346	RITESH	55000	4085
33346	TARUN	55000	4085
45209	NEHA	45000	4362
65320	VERONICA	100000	7361
65321	TINA	65300	4085
30270	VIVEK	43000	4362



45209	NEHA	45000	4362
65320	VERONICA	100000	7361
65321	TINA	65300	4085
30270	VIVEK	43000	4362
63705	KRITI	25000	7361
13640	JESSICA	65000	4362
84536	TANMAY	50000	4065
75446	MARK	70000	4050
75346	CHIRAG	55000	4050
96452	RITHIK	30000	4085
32645	TANISH	29000	4362
52600	VIJAY	35000	4362
15962	TIM	55000	7361
29043	EMILY	65000	7361

Download CSV

23 rows selected.

```

INSERT INTO DEP_MANAGER VALUES(84706, 4065);

INSERT INTO DEP_MANAGER VALUES(75446, 4050);

INSERT INTO DEP_MANAGER VALUES(25346, 6523);

INSERT INTO DEP_MANAGER VALUES(42346, 4085);

INSERT INTO DEP_MANAGER VALUES(13640, 4362); INSERT INTO

DEP_MANAGER VALUES(65320, 7361);

SELECT * FROM DEP_MANAGER;

```

```

74 INSERT INTO DEP_MANAGER VALUES(75446, 4050);
75 INSERT INTO DEP_MANAGER VALUES(25346, 6523);
76 INSERT INTO DEP_MANAGER VALUES(42346, 4085);
77 INSERT INTO DEP_MANAGER VALUES(13640, 4362);
78 INSERT INTO DEP_MANAGER VALUES(65320, 7361);
79
80 SELECT * FROM DEP_MANAGER;
81

```

DEP_MANAGER_ID	DEP_ID
75446	4050
25346	6523
65320	7361
42346	4085
84706	4065
13640	4362

[Download CSV](#)

6 rows selected.

INSERT INTO PROJECT VALUES(152, 'Android App for Hotel', 45209);

INSERT INTO PROJECT VALUES(201, 'Website development for retail store', 55362); INSERT INTO

PROJECT VALUES(124, 'IOS app for school', 15962);

SELECT \* FROM PROJECT;

```

94 INSERT INTO PROJECT VALUES(152, 'Android App for Hotel', 45209);
95 INSERT INTO PROJECT VALUES(201, 'Website development for retail store', 55362);
96 INSERT INTO PROJECT VALUES(124, 'IOS app for school', 15962);
97
98 SELECT * FROM PROJECT;
99

```

PROJECT_ID	PROJECT_TITLE	PRJ_MANAGER_ID
152	Android App for Hotel	45209
124	IOS app for school	15962
201	Website development for retail store	55362

[Download CSV](#)

3 rows selected.

-- Testing Skills

INSERT INTO EMPLOYEE\_SKILLS VALUES('MANUAL AND AUTOMATIC TESTING', 84536);

INSERT INTO EMPLOYEE\_SKILLS VALUES('AUTOMATIC TESTING, BUGZILLA, ZIRA TOOLS', 84706);

INSERT INTO EMPLOYEE\_SKILLS VALUES('SDLC, API AND GUI TESTING', 65034);

-- Software Dev Skills

INSERT INTO EMPLOYEE\_SKILLS VALUES('SQL, DSA, PYTHON', 75346);

INSERT INTO EMPLOYEE\_SKILLS VALUES('JAVA, SQL, R, GIT', 25634);

INSERT INTO EMPLOYEE\_SKILLS VALUES('C++, PYTHON, JAVA, DB', 75446);

INSERT INTO EMPLOYEE\_SKILLS VALUES('JAVA, R, LINUX, AWS', 55362);

-- Finance Skills

INSERT INTO EMPLOYEE\_SKILLS VALUES('COST ANALYSIS, JOURNAL ENTRY, RESILIENCE', 42346);

INSERT INTO EMPLOYEE\_SKILLS VALUES('BUFGETING, DATA PROCESSING, TAX PLANING ', 33346);

INSERT INTO EMPLOYEE\_SKILLS VALUES('TAX FILING, QUANTITATIVE AND RISK ANALYSIS', 96452);

INSERT INTO EMPLOYEE\_SKILLS VALUES('COST REDUCTION, CASH FLOW  
MANAGEMENT', 65321);

-- Customer Support Skills

INSERT INTO EMPLOYEE\_SKILLS VALUES('Persuasive communication, adaptibility', 44346);

INSERT INTO EMPLOYEE\_SKILLS VALUES('Self control, Assertiveness, Conflict Resolution ', 25346);

INSERT INTO EMPLOYEE\_SKILLS VALUES('Depersonization, Adaptibilty, Communication skills', 42316);

-- IOS dev Skills

INSERT INTO EMPLOYEE\_SKILLS VALUES('SWIFT, UI, DB',65320);

INSERT INTO EMPLOYEE\_SKILLS VALUES('SWIFT, XCODE, UI',15962);

INSERT INTO EMPLOYEE\_SKILLS VALUES('SWIFT, JSON, JAVA',63705); INSERT INTO

EMPLOYEE\_SKILLS VALUES('SWIFT, JAVA, XML',29043); -- Android dev Skills

INSERT INTO EMPLOYEE\_SKILLS VALUES('JAVA, XML, POSTGRESQL',32645);

INSERT INTO EMPLOYEE\_SKILLS VALUES('JAVA, XML, MATERIAL DESIGN',30270);

INSERT INTO EMPLOYEE\_SKILLS VALUES('JAVA, APIS, ANDROID SDK',13640);

INSERT INTO EMPLOYEE\_SKILLS VALUES('KOTLIN, DB, MATERIAL DESIGN',52600); INSERT INTO

EMPLOYEE\_SKILLS VALUES('KOTLIN, JAVA, XML, APIS',45209);

SELECT \* FROM EMPLOYEE\_SKILLS;

137	SELECT * FROM EMPLOYEE_SKILLS;
138	
SKILL_NAMES	EMP_ID
MANUAL AND AUTOMATIC TESTING	84536
JAVA, SQL, R, GIT	25634
C++, PYTHON, JAVA, DB	75446
COST ANALYSIS, JOURNAL ENTRY, RESILIENCE	42346
SWIFT, UI, DB	65320
SWIFT, JAVA, XML	29043
JAVA, XML, POSTGRESQL	32645
JAVA, XML, MATERIAL DESIGN	30270

TAX FILING, QUANTITATIVE AND RISK ANALYSIS	96452
Depersonization, Adaptibility, Communication skills	42316
SDLC, API AND GUI TESTING	65034
BUFGETING, DATA PROCESSING, TAX PLANING	33346
KOTLIN, JAVA, XML, APIS	45209
AUTOMATIC TESTING, BUGZILLA, ZIRA TOOLS	84706
SQL, DSA, PYTHON	75346
JAVA, R, LINUX, AWS	55362
COST REDUCTION, CASH FLOW MANAGEMENT	65321
Persuasive communication, adaptibility	44346
Self control, Assertiveness, Conflict Resolution	25346
SWIFT, XCODE, UI	15962
SWIFT, JSON, JAVA	63705
KOTLIN, DB, MATERIAL DESIGN	52600

Download CSV

23 rows selected.

-- Android project employees

INSERT INTO WORKS\_ON VALUES(152, 44346);

INSERT INTO WORKS\_ON VALUES(152, 52600);

INSERT INTO WORKS\_ON VALUES(152, 45209);

INSERT INTO WORKS\_ON VALUES(152, 32645);

INSERT INTO WORKS\_ON VALUES(152, 30270);

INSERT INTO WORKS\_ON VALUES(152, 96452);

INSERT INTO WORKS\_ON VALUES(152, 84536);

-- IOS project employees

INSERT INTO WORKS\_ON VALUES(124, 63705 );

INSERT INTO WORKS\_ON VALUES(124, 29043);

INSERT INTO WORKS\_ON VALUES(124, 15962);

INSERT INTO WORKS\_ON VALUES(124, 42316);

INSERT INTO WORKS\_ON VALUES(124, 33346);

INSERT INTO WORKS\_ON VALUES(124, 65034);

-- School website project employee

INSERT INTO WORKS\_ON VALUES(201, 75346);

INSERT INTO WORKS\_ON VALUES(201, 55362);

INSERT INTO WORKS\_ON VALUES(201, 25634);

INSERT INTO WORKS\_ON VALUES(201, 63705);

INSERT INTO WORKS\_ON VALUES(201, 44346);

INSERT INTO WORKS\_ON VALUES(201, 52600);

INSERT INTO WORKS\_ON VALUES(201, 33346); INSERT INTO  
WORKS\_ON VALUES(201, 65034);

SELECT \* FROM WORKS\_ON;

```
173 SELECT * FROM WORKS_ON;
174
```

PROJECT_ID	EMP_ID
124	15962
124	29043
124	33346
124	42316
124	63705
124	65034
152	30270
152	32645
152	44346
152	45209
152	52600

152	45209
152	52600
152	84536
152	96452
201	25634
201	33346
201	44346
201	52600
201	55362
201	63705
201	65034
201	75346

[Download CSV](#)

21 rows selected.

## EXECUTION - SQL Queries

- 1) **Write an SQL query to count employees by project.**

```
SELECT PROJECT.PROJECT_TITLE AS 'PROEJECT TITLE',
COUNT(WORKS_ON.EMP_ID) AS 'COUNT'FROM PROJECT, WORKS_ON WHERE
PROJECT.PROJECT_ID = WORKS_ON.PROJECT_ID GROUP BY
(WORKS_ON.PROJECT_ID);
```

Output:

	PROEJECT TITLE	COUNT
▶	IOS app for school	6
	Android App for Hotel	7
	Website development for retail store	8

- 2) **Write an SQL query to display the employee names whose name starts with 'V' and works in 'Android dev department'.**

```
SELECT EMP_NAME FROM EMPLOYEE WHERE EMP_NAME LIKE 'V%' AND DEP_ID =
4362;
```

Output:

	EMP_NAME
▶	VIVEK
	VIJAY

- 3) **Write an SQL query to Display the employee names who possess 'SQL' skill.**

```
SELECT EMP_NAME FROM EMPLOYEE WHERE EMP_ID IN (SELECT EMP_ID FROM
EMPLOYEE_SKILLS WHERE SKILL_NAMES LIKE '%SQL%'); Output:
```

	EMP_NAME
▶	JOHN
	TANISH
	CHIRAG



- 4) **Write an SQL query to display the employee whose salary is above 40000 and poses 'JAVA' skill.**

```
SELECT EMPLOYEE.EMP_NAME FROM EMPLOYEE WHERE EMPLOYEE.SALARY >
40000 AND EMPLOYEE.EMP_ID IN (SELECT EMP_ID FROM EMPLOYEE_SKILLS
WHERE SKILL_NAMES LIKE '%JAVA%'); Output:
```

	EMP_NAME
▶	JESSICA
	JOHN
	EMILY
	VIVEK
	NEHA
	JAMES
	MARK

- 5) **Write an SQL query to display names of project managers who are from 'Software dev department.**

```
SELECT EMPLOYEE.EMP_NAME AS 'PROJECT MANAGER NAME',
PROJECT.PROJECT_TITLE AS 'PROJECT' FROM EMPLOYEE, PROJECT WHERE EMPLOYEE.EMP_ID =
PROJECT.PRJ_MANAGER_ID AND EMPLOYEE.DEP_ID = 4050;
```

Output:

	PROJECT MANAGER NAME	PROJECT
▶	JAMES	Website development for retail store

- 6) **Write an SQL query to display the department name whose average employee salary is highest.**

```
SELECT DEPARTMENT.DEP_NAME, AVG(EMPLOYEE.SALARY) FROM EMPLOYEE,
DEPARTMENT WHERE (DEPARTMENT.DEP_ID = EMPLOYEE.DEP_ID) GROUP BY
DEPARTMENT.DEP_NAME ORDER BY(AVG(EMPLOYEE.SALARY)) DESC LIMIT 0,1 ;
```

Output:

	DEP_NAME	AVG(EMPLOYEE.SALARY)
▶	SOFTWARE_DEVELOPMENT	68750

7) **Write an SQL query to fetch ID, name, dep name of employees who are not department managers**

```
SELECT DISTINCT EMPLOYEE.EMP_ID, EMPLOYEE.EMP_NAME,
DEPARTMENT.DEP_NAME FROM EMPLOYEE, DEPARTMENT, DEP_MANAGER
WHERE (EMPLOYEE.DEP_ID = DEPARTMENT.DEP_ID AND EMPLOYEE.EMP_ID NOT
IN (SELECT MGR_ID FROM DEP_MANAGER)) ; Output:
```

	EMP_ID	EMP_NAME	DEP_NAME
▶	63705	KRITI	IOS_Development
	29043	EMILY	IOS_Development
	15962	TIM	IOS_Development
	44346	RISHAB	CUSTOMER_SUPPORT
	42316	RAVI	CUSTOMER_SUPPORT
	52600	VIJAY	ANDROID_DEVELOPMENT
	45209	NEHA	ANDROID_DEVELOPMENT
	32645	TANISH	ANDROID_DEVELOPMENT
	30270	VIVEK	ANDROID_DEVELOPMENT
	96452	RITHIK	FINANCE_DEPARTMENT
	65321	TINA	FINANCE_DEPARTMENT
	33346	TARUN	FINANCE_DEPARTMENT
	84536	TANMAY	TESTING_TEAM
	65034	BEN	TESTING_TEAM
	75346	CHIRAG	SOFTWARE_DEVELOPMENT
	55362	JAMES	SOFTWARE_DEVELOPMENT
	25634	JOHN	SOFTWARE_DEVELOPMENT

8) **Write an SQL program to count employees by department**

```
SELECT DEPARTMENT.DEP_NAME AS 'DEPARTMENT NAME',
COUNT(EMPLOYEE.EMP_ID) AS 'COUNT' FROM EMPLOYEE, DEPARTMENT WHERE
(DEPARTMENT.DEP_ID = EMPLOYEE.DEP_ID) GROUP BY DEPARTMENT.DEP_NAME;
```

Output:

	DEPARTMENT NAME	COUNT
▶	SOFTWARE_DEVELOPMENT	4
	TESTING_TEAM	3
	FINANCE_DEPARTMENT	4
	ANDROID_DEVELOPMENT	5
	CUSTOMER_SUPPORT	3
	IOS_Development	4

- 9) **Write an SQL query to display employee names who belongs to IOS project and earns more than 50,000 rupees.**

SELECT EMP\_NAME, SALARY FROM EMPLOYEE WHERE DEP\_ID = 7361 AND SALARY > 50000; Output:

	EMP_NAME	SALARY
▶	TIM	55000
	EMILY	65000
	VERONICA	100000

- 10) **Write an SQL query to fetch the employee names who are not working in any project.alter**

SELECT EMP\_NAME FROM EMPLOYEE WHERE EMP\_ID NOT IN (SELECT EMP\_ID FROM WORKS\_ON) AND EMP\_ID NOT IN (SELECT DEP\_MANAGER\_ID FROM DEP\_MANAGER); Output:

	EMP_NAME
▶	TINA

## PL/SQL:

**Functions: Write an PL/SQL program to display maximum salary using fucntions.**

DECLARE

TYPE SALARY IS VARRAY(5) OF INTEGER;

SALARIES SALARY;

function maxsalary (n SALARY)

return integer

is

    i integer;

    maxm integer;

    total integer;

begin

    maxm := n(1);

    total := n.count;

    for i in 1..total loop

        if n(i) > maxm then

            maxm:= SALARY(i);

        END if;

    end loop; end;

BEGIN

    SALARIES := SALARY(54000, 45000, 60000, 24000, 74000);

    dbms\_output.put\_line('The Maximum salary is: ' || maxsalary(SALARIES));

END;

### **Output:**

The Maximum salary is 74000

**Procedure: Write an PL/SQL program to display maximum salary using procedure.**

DECLARE

TYPE SALARY IS VARRAY(5) OF INTEGER;

SALARIES SALARY; procedure maxsalary(n SALARY)

Is

    i integer;

    maxm integer;

    total integer;

begin

    maxm := n(1);

    total := n.count;

    for i in 1..total loop

        if n(i) > maxm then

            maxm:= SALARY(i);

        END if;

    end loop;

    print('The Maximum salary is ' || maxm)

end;

BEGIN

    SALARIES := SALARY(5400, 4500, 6000, 24000, 7400);

    maxsalary(SALARIES);

END;

**Output:** The Maximum salary is 74000

## SCHEMA REFINEMENT - Normalization

### First Normal Form (1NF):

The Rules of 1NF are:

- ☐ Each column should contain atomic values.
- ☐ It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- ☐ Each column should hold values that are of same type ?      Each column should have a unique name.
- ☐ Order in which data is saved in the table does not matter.
- ☐ First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Here, all tables follow all the above rules. No attribute of any table contains atomic values, each column has a unique name and holds same data type.

Therefore, all the tables are already in 1NF. So no alteration is required.

### Second Normal Form (2NF):

The Rules of 2NF are:

- ☐ The table should be in first normal form.
- ☐ It should not have any partial dependencies.

Here, all tables are in first normal form and no column is partially dependent on primary keys.

Therefore, all the tables are already in 2NF. So no alteration is required.

### Third Normal Form(3NF):

- ☐ The table should be in second normal form.
- ☐ It should not have any transitive dependencies.

Here, all tables are in second normal form and no column is transitive dependent.

Therefore, all the tables are already in 3NF. So no alteration is required.

### Identifying candidate key(s):

**Super Key:** Super key can be defined as a set of those keys that identify a row or a tuple uniquely. The word super means superiority of keys.

**Candidate key:** The candidate key is a minimal subset of super keys. where the key which contains no redundant attribute is none other than a Candidate Key. An attribute holding a candidate key can never be NULL, which means its values cannot be null.

#### 1. Table Department:

DEPARTMENT
DEP_ID
DEP_NAME

In the above department table, there is only one super key i.e. {Dep\_ID}

Therefore, the candidate key will also be the {Dep\_ID}.

#### 2. Table Employee:

EMPLOYEE
EMP_ID
EMP_NAME
SALARY
DEP ID

In the above table, Employee ID attribute is unique and it can be used to fetch other attributes.

The super key is {Emp\_ID}

And the minimal subset of the super key i.e. candidate key is {Emp\_ID}.

### 3. Table Dep\_Manager

DEP MANAGER
Dep_Manager_ID
DEP_ID

For the above table, the super keys are:

{Dep\_ID}, { Dep\_Manager\_ID}, { Dep\_ID , Dep\_Manager\_ID }

And the minimal subset of the super key i.e. candidate key is {Dep\_ID}, { Dep\_Manager\_ID }.

### 4. Table Projects:

PROJECT
PRJ_MANAGER_ID
PRJ_ID
PRJ_TITLE

For the above table, the super keys are:

{ PRJ\_MANAGER\_ID }, { PRJ\_ID }, { PRJ\_TITLE }, { PRJ\_MANAGER\_ID, PRJ\_ID }, { PRJ\_ID, PRJ\_MANAGER\_ID }, { PRJ\_MANAGER\_ID , PRJ\_TITLE }, { PRJ\_ID , PRJ\_TITLE }, { PRJ\_TITLE , PRJ\_ID, PRJ\_MANAGER\_ID }

And the minimal subset of the super key i.e. candidate key is { PRJ\_MANAGER\_ID }, { PRJ\_ID }, { PRJ\_TITLE }.



#### 5. Table Employee Skills:

Employee_SKILLS
EMP_ID
SKILL_Names

For the above table, the super key is {EMP\_ID}.

And the minimal subset of the super key i.e. candidate key is {EMP\_ID}

#### 6. Table Works\_ON:

WORKS ON
EMP_ID
PRJ_ID

For the above table, the super key is {EMP\_ID, PRJ\_ID}.

And the minimal subset of the super key i.e. candidate key is {EMP\_ID, PRJ\_ID}.

## Conclusion

This implementation can help in managing a database of a company. The company provides from a single application program through to complete installation of hardware with customized software.

## References

[1] Normalization from JavaTpoint.

Link: <https://www.javatpoint.com/dbms-normalization>

[2] Arrays in PL/SQL from TutorialsPoint

Link: [https://www.tutorialspoint.com/plsql/plsql\\_arrays.htm](https://www.tutorialspoint.com/plsql/plsql_arrays.htm)

**THE END THANK YOU**