Experiment 2

Student Name: Amit Kumar **UID: 23BCS12430**
Branch: BE CSE **Section/Group: KRG_1-A**
Semester: 5 **Date of Performance: 31-07-25**
Subject Name: ADBMS **Subject Code: 23CSP-333**

1. Aim:

Q.1. Organizational Hierarchy Explorer

You are a Database Engineer at TalentTree Inc., an enterprise HR analytics platform that stores employee data, including their reporting relationships. The company maintains a centralized Employee relation that holds: Each employee's ID, name, department, and manager ID (who is also an employee in the same table).

Your task is to generate a report that maps employees to their respective managers, showing: The employee's name and department, their manager's name and department (if applicable) This will help the HR department visualize the internal reporting hierarchy.

Q.2. Financial Forecast Matching with Fallback Strategy

You are a Data Engineer at FinSight Corp, a company that models Net Present Value (NPV) projections for investment decisions.Your system maintains two key datasets:

1. Year_tbl: Actual recorded NPV's of various financial instruments over different years: ID: Unique Financial instrument identifier, YEAR: Year of record, NPV: Net Present Value in that year

2. Queries_tbl: A list of instrument-year pairs for which stakeholders are requesting NPV values: ID: Financial instrument identifier, YEAR: Year of interest.

Find the NPV of each query from the Queries table. Return the output order by ID and Year in the sorted form. However, not all ID-YEAR combinations in the Queries table are present in the Year_tbl. If an NPV is missing for a requested combination, assume it to be 0 to maintain a consistent financial report.

2. Requirements (Hardware/Software):

Microsoft SQL server

3. Procedure:

Q.1. Code:

```
CREATE TABLE TBL_EMPLOYEE (
  empId int primary key,
  name varchar(15) not null,
  dept varchar(10) not null,
  managerId int null,
  foreign key(managerId) references TBL_EMPLOYEE(empId)
);
-- insert
INSERT INTO TBL_EMPLOYEE(empId,name,dept,managerId) VALUES (1, 'Clark',
'Sales', null);
INSERT INTO TBL_EMPLOYEE(empId,name,dept,managerId) VALUES (2, 'Dave',
'Accounting', 1);
INSERT INTO TBL_EMPLOYEE(empId,name,dept,managerId) VALUES (3, 'Ava',
'Accounting',1);
INSERT INTO TBL_EMPLOYEE(empId,name,dept,managerId) VALUES (4, 'Eve',
'Sales',2);
INSERT INTO TBL_EMPLOYEE(empId,name,dept,managerId) VALUES (5, 'Borris',
'Marketing',3);

select e.empId, e.name, ep.name, ep.dept from
TBL_EMPLOYEE as e
left outer join
TBL_EMPLOYEE as ep
on e.managerid=ep.empId;
```

Q.2. Code:

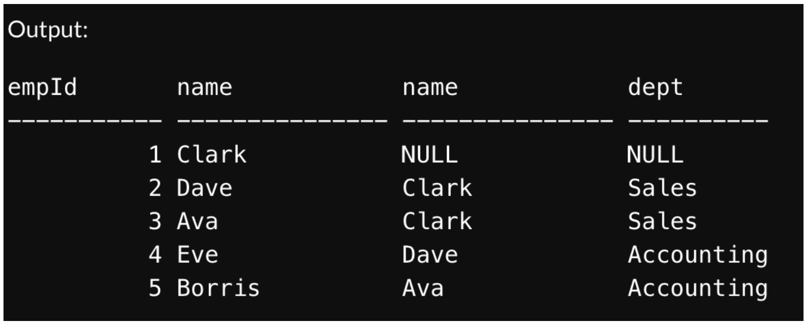
```
CREATE TABLE tbl_year(
 ID INT,
 YEAR INT,
 NPV INT
)
INSERT INTO tbl_year(ID, YEAR, NPV)
VALUES
(1,2018,100),
(7,2020,30),
(13,2019,40),
(1,2019,113),
(2,2008,121),
(3,2002,12),
(11,2020,99),
(7,2019,0);
```

```
CREATE TABLE tbl_query(
ID INT,
YEAR INT
);
INSERT INTO tbl_query(ID, YEAR)
VALUES
(1,2019),
(2,2008),
(3,2009),
(7,2018),
(7,2019),
(7,2020),
(13,2019);

select tbl_query.id, tbl_query.year, isnull(tbl_year.npv,0) as NPV from
tbl_query
left outer join
tbl_year
on tbl_year.id=tbl_query.id and tbl_year.year=tbl_query.year
order by tbl_query.id, tbl_query.year;
```

4. Output:

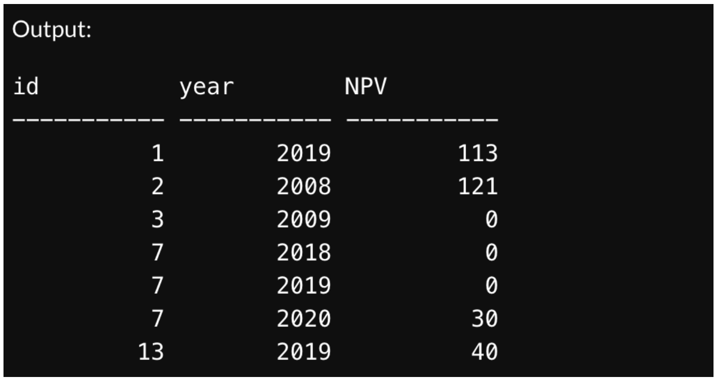
Q.1.



```
Output:

empId       name            name            dept
----------- --------------- --------------- ----------
          1 Clark           NULL            NULL
          2 Dave            Clark           Sales
          3 Ava             Clark           Sales
          4 Eve             Dave            Accounting
          5 Borris          Ava             Accounting
```

Q.2.



```
Output:

id          year        NPV
----------- ----------- -----------
          1        2019         113
          2        2008         121
          3        2009           0
          7        2018           0
          7        2019           0
          7        2020          30
         13        2019          40
```

5. Learning Outcome:

- **Understand the purpose and use cases of different SQL join types (INNER, LEFT, RIGHT, FULL).**

- **Apply SQL joins to retrieve data from multiple related tables effectively.**

- **Analyze the impact of join conditions on query results and data integrity.**

- **Construct optimized SQL queries using joins for real-world database scenarios.**