SQL_A2 Database Analysis

By Shanti Jogi
03-02-25

Project Overview:

This project involves creating a database named SQL_A2, importing three tables (DEPARTMENT_TB, EMPLOYEE_TB, and COMPANY_TB), and executing 20 SQL queries to extract meaningful insights.

Database Creation:

```
CREATE DATABASE SQL_A2;
USE SQL_A2;
```

Imported Tables:

1. DEPARTMENT_TB - Stores department details.
2. EMPLOYEE_TB - Stores employee details including salary and department ID.
3. COMPANY_TB - Stores company details including revenue and location.

Executed SQL Queries:

1. Query all rows from the DEPARTMENT_TB table:

```
SELECT * FROM DEPARTMENT_TB;
```

2. Change the name of the department with ID = 1 to 'Management':

```
UPDATE DEPARTMENT_TB SET NAME='Management'
WHERE ID = 1;
```

3. Delete employees with a salary greater than 100000:

```
DELETE FROM EMPLOYEE_TB
WHERE SALARY > 100000;
```

4. Query the names of all companies:

```
SELECT NAME
FROM COMPANY_TB;
```

5. Query the name and city of every employee:

```
SELECT NAME, CITY
FROM EMPLOYEE_TB;
```

6. Query all companies with revenue greater than 5000000:

```
SELECT NAME
FROM COMPANY_TB
WHERE REVENUE > 5000000;
```

7. Query all companies with revenue smaller than 5000000:

```
SELECT NAME
FROM COMPANY_TB
WHERE REVENUE < 5000000;
```

8. Query all companies with revenue smaller than 5000000 without using '<' operator:

```
SELECT NAME
FROM COMPANY_TB
WHERE REVENUE BETWEEN 0 AND 4999999;
```

9. Query all employees with salary between 50000 and 70000:

```
SELECT NAME
FROM EMPLOYEE_TB
WHERE SALARY BETWEEN 50000 AND 70000;
```

10. Query all employees with salary between 50000 and 70000 without using BETWEEN:

```
SELECT NAME
FROM EMPLOYEE_TB
WHERE SALARY >= 50000 AND SALARY <= 70000;
```

11. Query all employees with salary equal to 80000:

```
SELECT NAME
FROM EMPLOYEE_TB
WHERE SALARY = 80000;
```

12. Query all employees with salary not equal to 80000:

```
SELECT NAME
FROM EMPLOYEE_TB
WHERE SALARY <> 80000;
```

13. Query all unique department names:

```
SELECT DISTINCT NAME
FROM DEPARTMENT_TB;
```

14. Query names of employees with their department ID (without using JOIN):

```
SELECT NAME,
(SELECT ID FROM DEPARTMENT_TB WHERE DEPARTMENT_TB.ID =
EMPLOYEE_TB.DEPARTMENT_ID) AS DEPARTMENT_ID
FROM EMPLOYEE_TB;
```

15. Query names of employees with their department ID using JOIN:

```
SELECT A.NAME AS EMP_NAME, B.DEPARTMENT_ID AS DEP_ID
FROM EMPLOYEE_TB A
JOIN EMPLOYEE_TB B ON B.DEPARTMENT_ID = A.DEPARTMENT_ID;
```

16. Query each company's name together with each department:

```
SELECT C.NAME AS COMP_NAME, D.NAME AS DEPT_NAME
FROM COMPANY_TB C
CROSS JOIN (SELECT DISTINCT NAME FROM DEPARTMENT_TB) D;
```

17. Query employees with department names they are not working in :

```
SELECT E.NAME AS EMP_NAME, D.NAME AS DEPT_NAME
FROM EMPLOYEE_TB E
INNER JOIN DEPARTMENT_TB D  ON  D.ID<>E.DEPARTMENT_ID;
```

18. Query names of all companies with column name changed to 'Company':

```
SELECT NAME AS COMPANY
FROM COMPANY_TB;
OR
SELECT COMPANY
FROM ( SELECT NAME AS COMPANY  FROM COMPANY_TB ) AS  A;
```

19. Query total salary per city:

```
SELECT CITY, SUM(SALARY) AS TOTAL_SALARY
FROM EMPLOYEE_TB
GROUP BY CITY;
```

20. Query the highest revenue company:

```
SELECT NAME
FROM COMPANY_TB
ORDER BY REVENUE DESC
LIMIT 1;
OR
SELECT NAME
FROM COMPANY_TB
WHERE REVENUE = (SELECT MAX(REVENUE) FROM COMPANY_TB);
```

Conclusion:

- The queries performed include data retrieval, updates, deletions, and joins.
- Various SQL operations such as filtering, ordering, and aggregations were applied.
- The dataset provided insights into company revenue, employee salaries, department details, and city-wise salary distribution.

This project demonstrates fundamental SQL operations and data handling techniques using MySQL.