Company One

The employees table, called EMP, contains the attributes EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPNO:

EMPNO is a unique employee number, it is the primary key of the employee table.	ENAME stores the employee's name.	The JOB attribute stores the name of the job of the employee.	MGR contains the employee number of the employee who manages that employee, if the employee has no manager, then the MGR column for that employee is left set to null.
HIREDATE stores the date on which the employee joined the Company.	SAL contains the details of employee salaries.	COMM stores values of commission paid to employees, not all employees receive commission, in which case the COMM field is set to null.	DEPTNO stores the department number of the department in which each employee is based. This data item acts as a foreign key, linking the employee details stored in the EMP table with the details of departments in which employees work, which are stored in the DEPT table.

The DEPT table contains three columns:

DEPTNO is the primary key containing the department numbers used to identify each department.	DNAME is the name of each department.
LOC is the location where each department is based.	

Tools Used:

https://onecompiler.com/ https://pinetools.com/syntax-highlighter https://mermaid.live/

DDL code use to create the database and tables

```
-- Create the database
CREATE DATABASE CompanyDB;

-- Use the created database
USE CompanyDB;

-- Create the DEPT table
CREATE TABLE DEPT (
    DEPTNO INT PRIMARY KEY,          -- DEPTNO is the primary key of the DEPT table
    DNAME VARCHAR(20),               -- DNAME stores the name of the department
    LOC VARCHAR(20)                  -- LOC stores the location of the department
);

-- Create the EMP table
CREATE TABLE EMP (
    EMPNO INT PRIMARY KEY,           -- EMPNO is the primary key of the EMP table
    ENAME VARCHAR(20),               -- ENAME stores the employee's name
    JOB VARCHAR(20),                 -- JOB stores the name of the job of the employee
    MGR INT,                         -- MGR stores the manager's EMPNO, can be NULL if no manager
    HIREDATE DATE,                   -- HIREDATE stores the date the employee was hired
    SAL DECIMAL(10, 2),              -- SAL stores the employee's salary
    COMM DECIMAL(10, 2),             -- COMM stores the employee's commission, can be NULL
    DEPTNO INT,                      -- DEPTNO stores the department number, acts as a foreign key
    FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO) -- Foreign key links EMP's DEPTNO to DEPT table
);
```

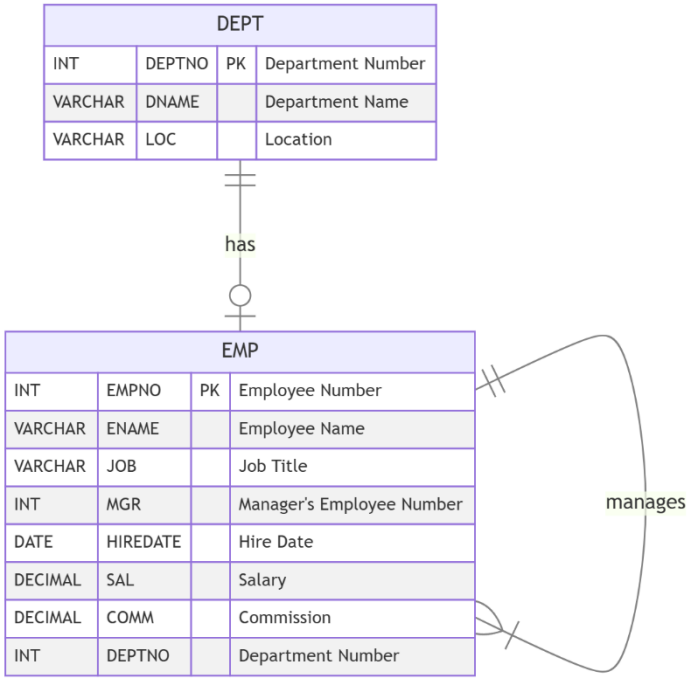
DML code used to add the data to the tables

```
INSERT INTO DEPT (DEPTNO, DNAME, LOC) VALUES
(10, 'ACCOUNTING', 'NEW YORK'),
(20, 'RESEARCH', 'DALLAS'),
(30, 'SALES', 'CHICAGO');

INSERT INTO EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO) VALUES
(7369, 'SMITH', 'CLERK', 7902, '1980-12-17', 800.00, NULL, 20),
(7499, 'ALLEN', 'SALESMAN', 7698, '1981-02-20', 1600.00, 300.00, 30),
(7521, 'WARD', 'SALESMAN', 7698, '1981-02-22', 1250.00, 500.00, 30),
(7566, 'JONES', 'MANAGER', 7839, '1981-04-02', 2975.00, NULL, 20),
(7654, 'MARTIN', 'SALESMAN', 7698, '1981-09-28', 1250.00, 1400.00, 30),
(7698, 'BLAKE', 'MANAGER', 7839, '1981-05-01', 2850.00, NULL, 30),
(7782, 'CLARK', 'MANAGER', 7839, '1981-06-09', 2450.00, NULL, 10),
(7788, 'SCOTT', 'ANALYST', 7566, '1987-04-19', 3000.00, NULL, 20),
(7839, 'KING', 'PRESIDENT', NULL, '1981-11-17', 5000.00, NULL, 10),
(7844, 'TURNER', 'SALESMAN', 7698, '1981-09-08', 1500.00, 0.00, 30),
(7876, 'ADAMS', 'CLERK', 7788, '1987-05-23', 1100.00, NULL, 20),
(7900, 'JAMES', 'CLERK', 7698, '1981-12-03', 950.00, NULL, 30),
(7902, 'FORD', 'ANALYST', 7566, '1981-12-03', 3000.00, NULL, 20),
(7934, 'MILLER', 'CLERK', 7782, '1982-01-23', 1300.00, NULL, 10);


SELECT * FROM DEPT;
SELECT * FROM EMP;
```

Tables



Department

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

Employee

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-04-02	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850.00	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450.00	NULL	10
7788	SCOTT	ANALYST	7566	1987-04-19	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	1981-11-17	5000.00	NULL	10
7844	TURNER	SALESMAN	7698	1981-09-08	1500.00	0.00	30
7876	ADAMS	CLERK	7788	1987-05-23	1100.00	NULL	20
7900	JAMES	CLERK	7698	1981-12-03	950.00	NULL	30
7902	FORD	ANALYST	7566	1981-12-03	3000.00	NULL	20
7934	MILLER	CLERK	7782	1982-01-23	1300.00	NULL	10

1. List all Employees whose salary is greater than 1,000 but not 2,000. Show the Employee Name, Department and Salary (4 marks)

```
SELECT EMP.ENAME, DEPT.DNAME, EMP.SAL
FROM EMP
JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO
WHERE EMP.SAL > 1000 AND EMP.SAL < 2000;
```

```
+--------+------------+---------+
| ENAME  | DNAME      | SAL     |
+--------+------------+---------+
| MILLER | ACCOUNTING | 1300.00 |
| ADAMS  | RESEARCH   | 1100.00 |
| ALLEN  | SALES      | 1600.00 |
| WARD   | SALES      | 1250.00 |
| MARTIN | SALES      | 1250.00 |
| TURNER | SALES      | 1500.00 |
+--------+------------+---------+
```

The query selects the employee's name (ENAME) from the EMP table, the department name (DNAME) from the DEPT table, and the salary (SAL) from the EMP table to include in the result set. The main table being queried is EMP, and it is joined with the DEPT table using an inner join, linking the department number (DEPTNO) in both tables to ensure each employee is matched with their corresponding department. The WHERE clause ensures that only employees with a salary greater than 1,000 but less than 2,000 are included, while excluding those with salaries exactly equal to 1,000 or 2,000.

This could also be accomplished with between. The question appears not to be inclusive so the code would be '`WHERE EMP.SAL BETWEEN 1001 AND 1999;'` The operator method was chosen as they are not inclusive, meeting the requirements. It should also be mentioned that the while both approaches are acceptable, the between command can be less efficient with larger data sets.

Including the use of aliases would provide a more user friendly output (w3, 2024). By using aliases for the column names (such as "Employee Name" for EMP.ENAME), the query output is more intuitive for users who may not be familiar with the raw database schema.

```
SELECT EMP.ENAME AS "Employee Name",
       DEPT.DNAME AS "Department Name",
       EMP.SAL AS "Salary"
FROM EMP
JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO
WHERE EMP.SAL > 1000 AND EMP.SAL < 2000;
```

```
+---------------+-----------------+---------+
| Employee Name | Department Name | Salary  |
+---------------+-----------------+---------+
| MILLER        | ACCOUNTING      | 1300.00 |
| ADAMS         | RESEARCH        | 1100.00 |
| ALLEN         | SALES           | 1600.00 |
| WARD          | SALES           | 1250.00 |
| MARTIN        | SALES           | 1250.00 |
| TURNER        | SALES           | 1500.00 |
+---------------+-----------------+---------+
```

2. Count the number of people in department 30 who receive a salary and a commission. (4 marks)

```
SELECT COUNT(*) AS EmployeeCount
FROM EMP
WHERE DEPTNO = 30 AND SAL IS NOT NULL AND COMM IS NOT NULL;
```

```
+---------------+
| EmployeeCount |
+---------------+
|             4 |
+---------------+
```

The query counts the number of employees in department 30 who have both a non-null salary (SAL) and a non-null commission (COMM). It selects from the EMP table, applying a filter with the WHERE clause to restrict the results to employees in department 30 (DEPTNO = 30). The SAL IS NOT NULL condition ensures that only employees with a defined salary are included, and the COMM IS NOT NULL condition ensures that only employees with a defined commission are counted. The query then returns the total count of such employees under the alias EmployeeCount.

The original query works but does not consider the edge case of '0'. As Celko (2010) points out, NULL represents the absence of a value where 0 is a legitimate numeric value. The query can be improved by filtering for values greater than 0. 'SAL > 0' filters out employees with a salary of 0 or less. 'COMM > 0' filters out employees with a commission of 0 or less. This ensuring that only employees with a positive salary and with a positive commission are counted.

```
SELECT COUNT(*) AS EmployeeCount
FROM EMP
WHERE DEPTNO = 30
  AND SAL > 0
  AND COMM > 0;
```

```
+---------------+
| EmployeeCount |
+---------------+
|             3 |
+---------------+
```

3. Find the name and salary of the employees that have a salary greater or equal to 1,000 and live in Dallas. (4 marks)

```
SELECT E.ENAME AS "Employee Name", E.SAL AS "Salary"
FROM EMP E
INNER JOIN DEPT D ON E.DEPTNO = D.DEPTNO
WHERE E.SAL >= 1000 AND D.LOC = 'DALLAS';
```

```
+---------------+---------+
| Employee Name | Salary  |
+---------------+---------+
| JONES         | 2975.00 |
| SCOTT         | 3000.00 |
| ADAMS         | 1100.00 |
| FORD          | 3000.00 |
+---------------+---------+
```

The query retrieves the names and salaries of employees who earn a salary greater than or equal to 1,000 and work in Dallas. It selects the employee name (ENAME) and salary (SAL) from the EMP table, joining it with the DEPT table based on the department number (DEPTNO). The query uses the WHERE clause to filter employees whose salary meets the condition EMP.SAL >= 1000 and who work in the Dallas department.

A another approach would be using the EXISTS operator. This operator checks if a related department row exists with the condition. An advantage of this method is efficiency. It stops processing as soon as a match is found, making it ideal for cases where the existence of related data is the only concern. The EXISTS command is effective for specific existence checking tasks, but its suitability depends on the query context and dataset size.

```
SELECT E.ENAME AS "Employee Name", E.SAL AS "Salary"
FROM EMP E
WHERE E.SAL >= 1000
AND EXISTS (SELECT 1 FROM DEPT D WHERE D.DEPTNO = E.DEPTNO AND D.LOC = 'DALLAS');
```

```
+---------------+---------+
| Employee Name | Salary  |
+---------------+---------+
| JONES         | 2975.00 |
| SCOTT         | 3000.00 |
| ADAMS         | 1100.00 |
| FORD          | 3000.00 |
+---------------+---------+
```

4. Find all departments that do not have any current employees. (4 marks)

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO NOT IN (SELECT DISTINCT DEPTNO FROM EMP);
```

No Output

The query identifies department names (DNAME) from the DEPT table where the department number (DEPTNO) is not present in the EMP table, indicating departments with no employees. The subquery (SELECT DISTINCT DEPTNO FROM EMP) generates a distinct list of department numbers referenced in the EMP table. The NOT IN condition then filters the DEPT table to exclude departments with numbers found in this list. This approach is straightforward and ensures the result only includes departments without employees.

The query using NOT IN returned no results, which could indicate that all departments have employees. Given that there is no indication of employees leaving in the dataset, the revised query was written to explicitly determine whether any department lacks employees. By using the EXISTS operator within a CASE statement, the new SQL provides a definitive output. It confirms if departments without employees exist or states that all departments have employees. This approach ensures clarity and accounts for the dataset's limitations.

```
SELECT
    CASE
        WHEN EXISTS (
            SELECT 1
            FROM DEPT
            WHERE DEPTNO NOT IN (SELECT DISTINCT DEPTNO FROM EMP)
        )
        THEN 'There are departments without employees.'
        ELSE 'All departments have employees.'
    END AS Result;
```

```
+---------------------------------+
| Result                          |
+---------------------------------+
| All departments have employees. |
+---------------------------------+
```

Changes to the data set would address issues related to determining current employees. An END_DATE field could be added to track when an employee leaves the organisation, with NULL indicating they are still employed. This provides clarity on employment periods and allows for historical analysis. An EMPLOYED Boolean field could explicitly denote whether an employee is currently active (TRUE) or not (FALSE). This simplifies queries for current employees.

5. List the department number, the average salary, and the number/count of employees of each department. (4 marks)

```
SELECT EMP.DEPTNO,
    FORMAT(AVG(EMP.SAL), 2) AS AverageSalary,
    COUNT(*) AS EmployeeCount
FROM EMP
GROUP BY EMP.DEPTNO;
```

```
+--------+---------------+---------------+
| DEPTNO | AverageSalary | EmployeeCount |
+--------+---------------+---------------+
|     10 | 2,916.67      |             3 |
|     20 | 2,175.00      |             5 |
|     30 | 1,566.67      |             6 |
+--------+---------------+---------------+
```

The query retrieves the department number (DEPTNO), the average salary (AverageSalary), and the number of employees (EmployeeCount) for each department from the EMP table. It uses the AVG(EMP.SAL) function to calculate the average salary for each department and the COUNT(*) function to count the total number of employees in each department. The results are grouped by department number with the GROUP BY EMP.DEPTNO clause. The FORMAT(AVG(EMP.SAL), 2) function ensures that the average salary is displayed with two decimal places. This query helps to summarise the salary distribution and the employee count across different departments within the organisation.

SQL Code

```sql
-- Question One
SELECT EMP.ENAME AS "Employee Name",
       DEPT.DNAME AS "Department Name",
       EMP.SAL AS "Salary"
FROM EMP
JOIN DEPT ON EMP.DEPTNO = DEPT.DEPTNO
WHERE EMP.SAL > 1000 AND EMP.SAL < 2000;


-- Question Two
SELECT COUNT(*) AS EmployeeCount
FROM EMP
WHERE DEPTNO = 30
  AND SAL > 0
  AND COMM > 0;

-- Question Three
SELECT E.ENAME AS "Employee Name", E.SAL AS "Salary"
FROM EMP E
INNER JOIN DEPT D ON E.DEPTNO = D.DEPTNO
WHERE E.SAL >= 1000 AND D.LOC = 'DALLAS';

-- Question Four
SELECT
  CASE
    WHEN EXISTS (
      SELECT 1
      FROM DEPT
      WHERE DEPTNO NOT IN (SELECT DISTINCT DEPTNO FROM EMP)
    )
    THEN 'There are departments without employees.'
    ELSE 'All departments have employees.'
  END AS Result;

-- Question Five
SELECT EMP.DEPTNO,
    FORMAT(AVG(EMP.SAL), 2) AS AverageSalary,
    COUNT(*) AS EmployeeCount
FROM EMP
GROUP BY EMP.DEPTNO;
```

References

Celko’s, J. (2010) *Joe Celko’s SQL for Smarties, 4th Edition*. San Francisco, CA: Morgan Kaufmann.

Mermaid Live Editor (no date) *Mermaid Live Editor*. Available at: https://mermaid.live/ (Accessed: January 12, 2025).

One Compiler (no date) *One Compiler*. Available at: https://onecompiler.com/mysql (Accessed: January 12, 2025).

Pine Tools (no date) *Online Syntax highlighter*. Available at: https://pinetools.com/syntax-highlighter (Accessed: January 12, 2025).

UoEO (no date) *Assignment 1: Part 3 of 3*. Available at: https://www.my-course.co.uk/mod/assign/view.php?id=1077891 (Accessed: January 12, 2025).

W3 Schools (no date) *MySQL Aliases*. Available at: https://www.w3schools.com/mysql/mysql_alias.asp (Accessed: January 12, 2025).