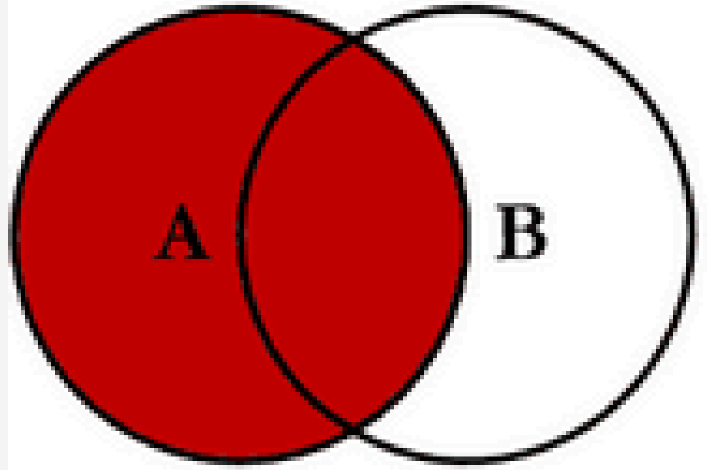
Add Company Name

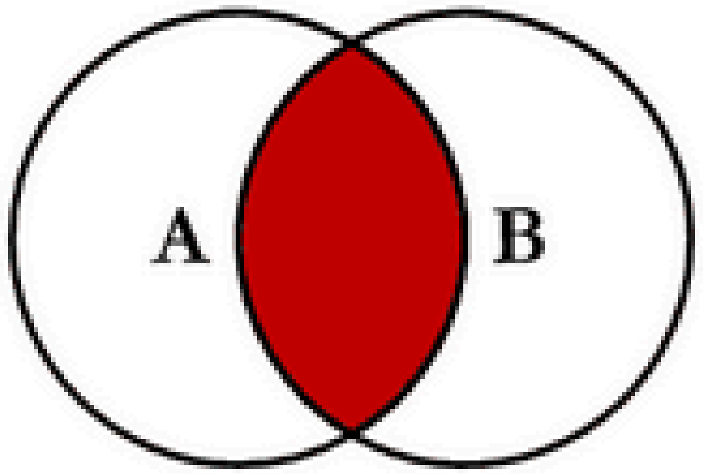
Tutorial 4

Hemang & Tejas

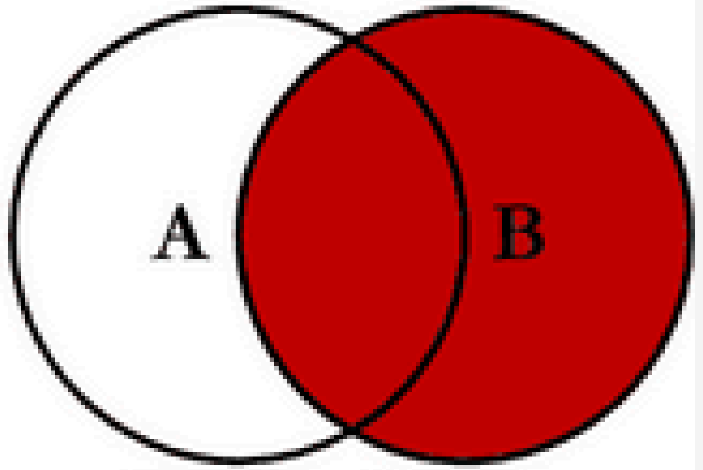
SQL JOINS



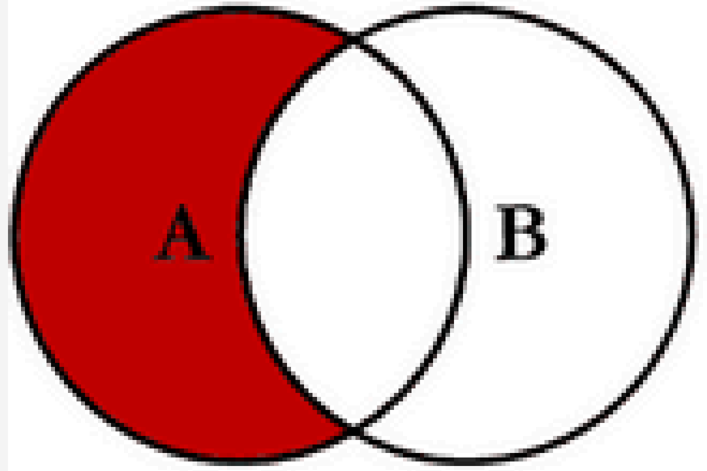
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



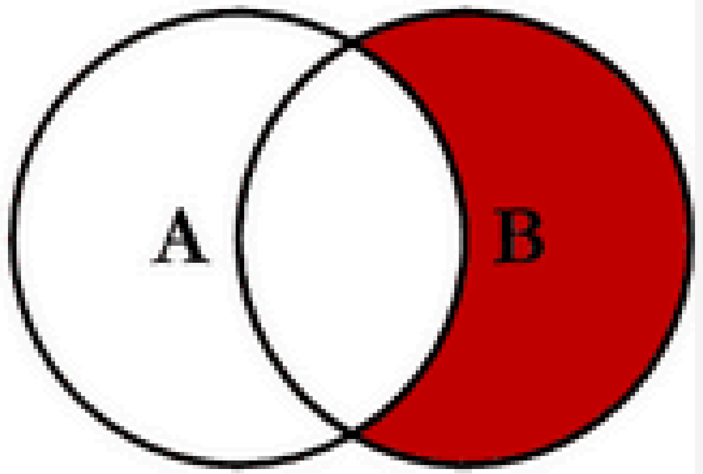
```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



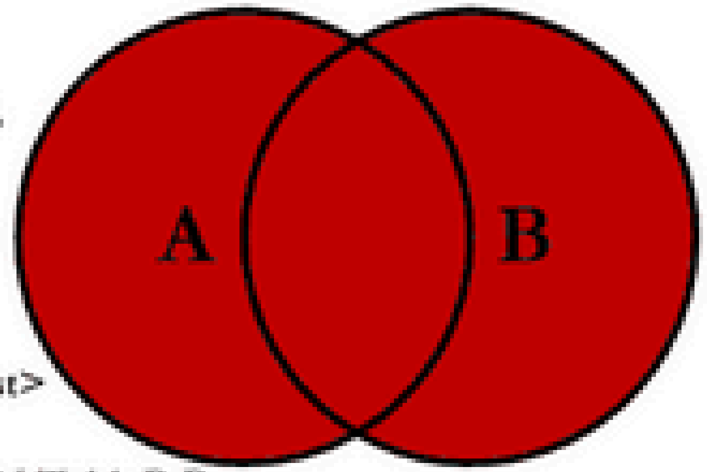
```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



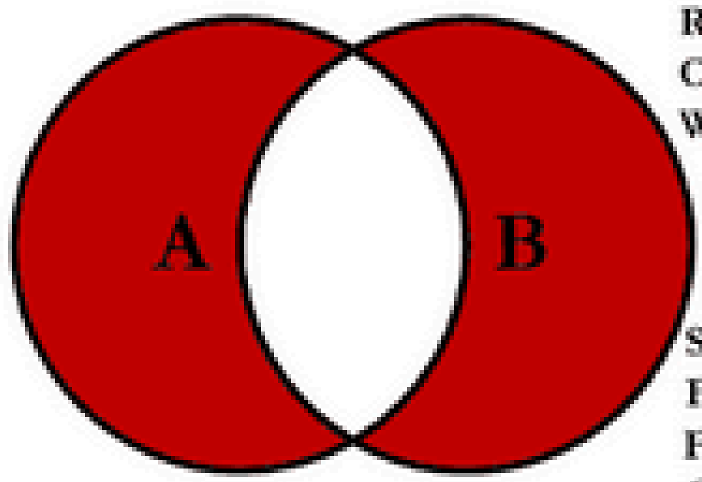
```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```



```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

INNER JOIN OR JOIN

The INNER JOIN keyword selects records that have matching values in both tables.

The INNER JOIN keyword returns only rows with a match in both tables.

Which means that if you have a Department with no Mgr_ssn, or with a Mgr_ssn that is not present in the EMPLOYEE table, that record would not be returned in the result.

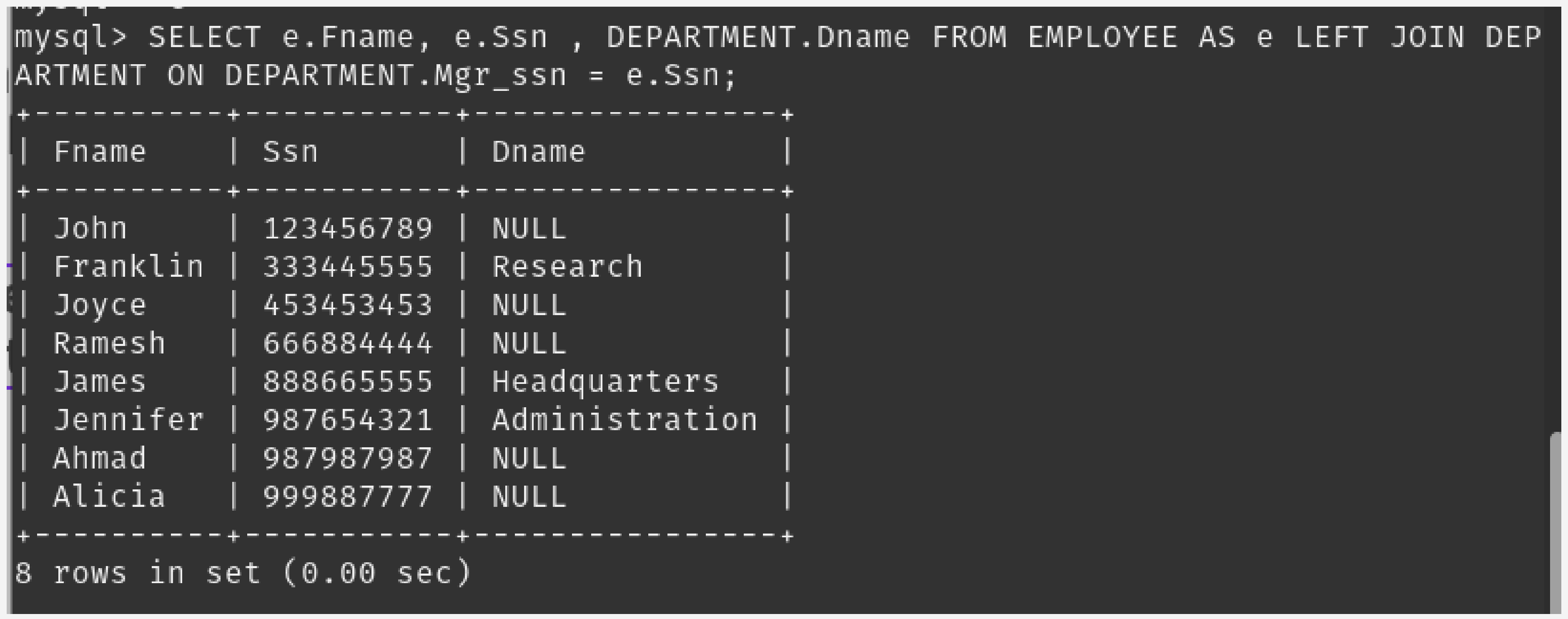
Eg : SELECT e.Fname, e.Ssn , DEPARTMENT.Dname FROM EMPLOYEE AS e INNER JOIN DEPARTMENT ON DEPARTMENT.Mgr_ssn = e.Ssn;

LEFT JOIN OR LEFT OUTER JOIN

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2).

The LEFT JOIN keyword returns all records from the left table (Customers), even if there are no matches in the right table (Orders).

Eg : SELECT e.Fname, e.Ssn , DEPARTMENT.Dname FROM EMPLOYEE AS e LEFT JOIN DEPARTMENT ON DEPARTMENT.Mgr_ssn = e.Ssn;



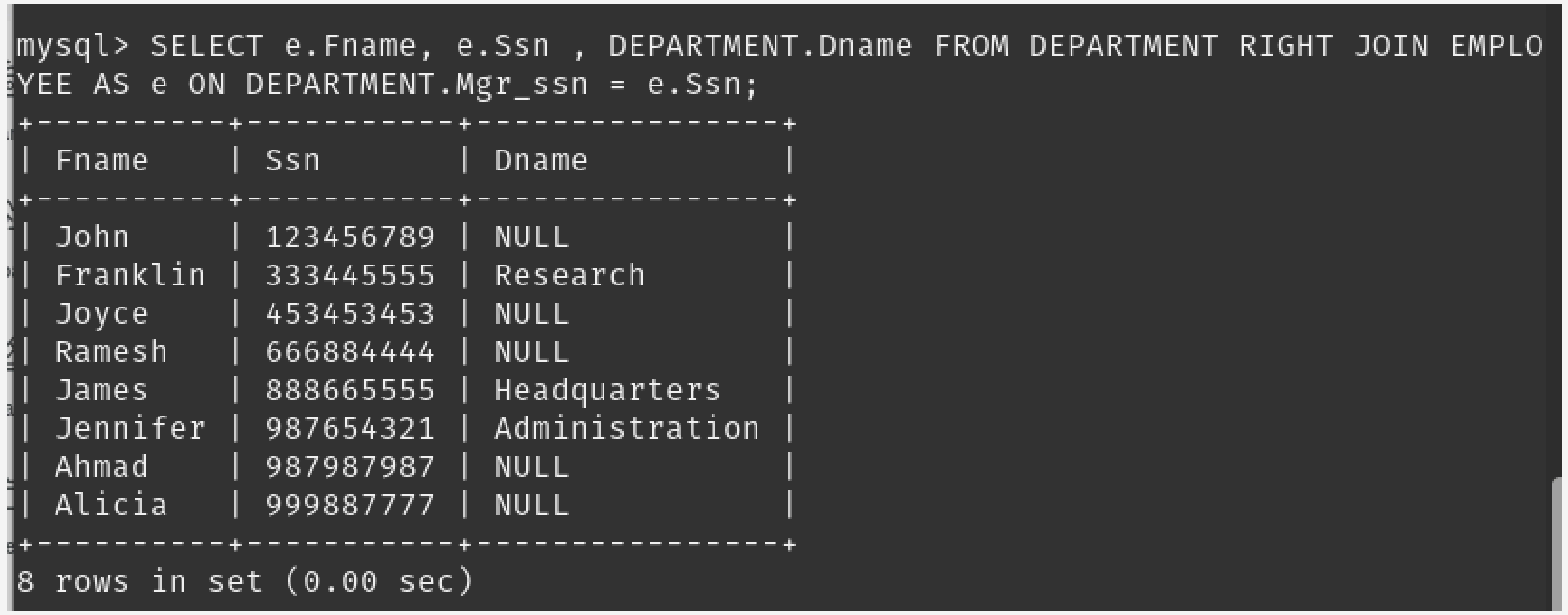
```
mysql> SELECT e.Fname, e.Ssn , DEPARTMENT.Dname FROM EMPLOYEE AS e LEFT JOIN DEP
ARTMENT ON DEPARTMENT.Mgr_ssn = e.Ssn;
+----------+-----------+----------------+
| Fname    | Ssn       | Dname          |
+----------+-----------+----------------+
| John     | 123456789 | NULL           |
| Franklin | 333445555 | Research       |
| Joyce    | 453453453 | NULL           |
| Ramesh   | 666884444 | NULL           |
| James    | 888665555 | Headquarters   |
| Jennifer | 987654321 | Administration |
| Ahmad    | 987987987 | NULL           |
| Alicia   | 999887777 | NULL           |
+----------+-----------+----------------+
8 rows in set (0.00 sec)
```

RIGHT JOIN OR RIGHT OUTER JOIN

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1).

The RIGHT JOIN keyword returns all records from the right table (Employees), even if there are no matches in the left table (Orders).

Eg : SELECT e.Fname, e.Ssn , DEPARTMENT.Dname FROM DEPARTMENT RIGHT JOIN EMPLOYEE AS e ON DEPARTMENT.Mgr_ssn = e.Ssn;



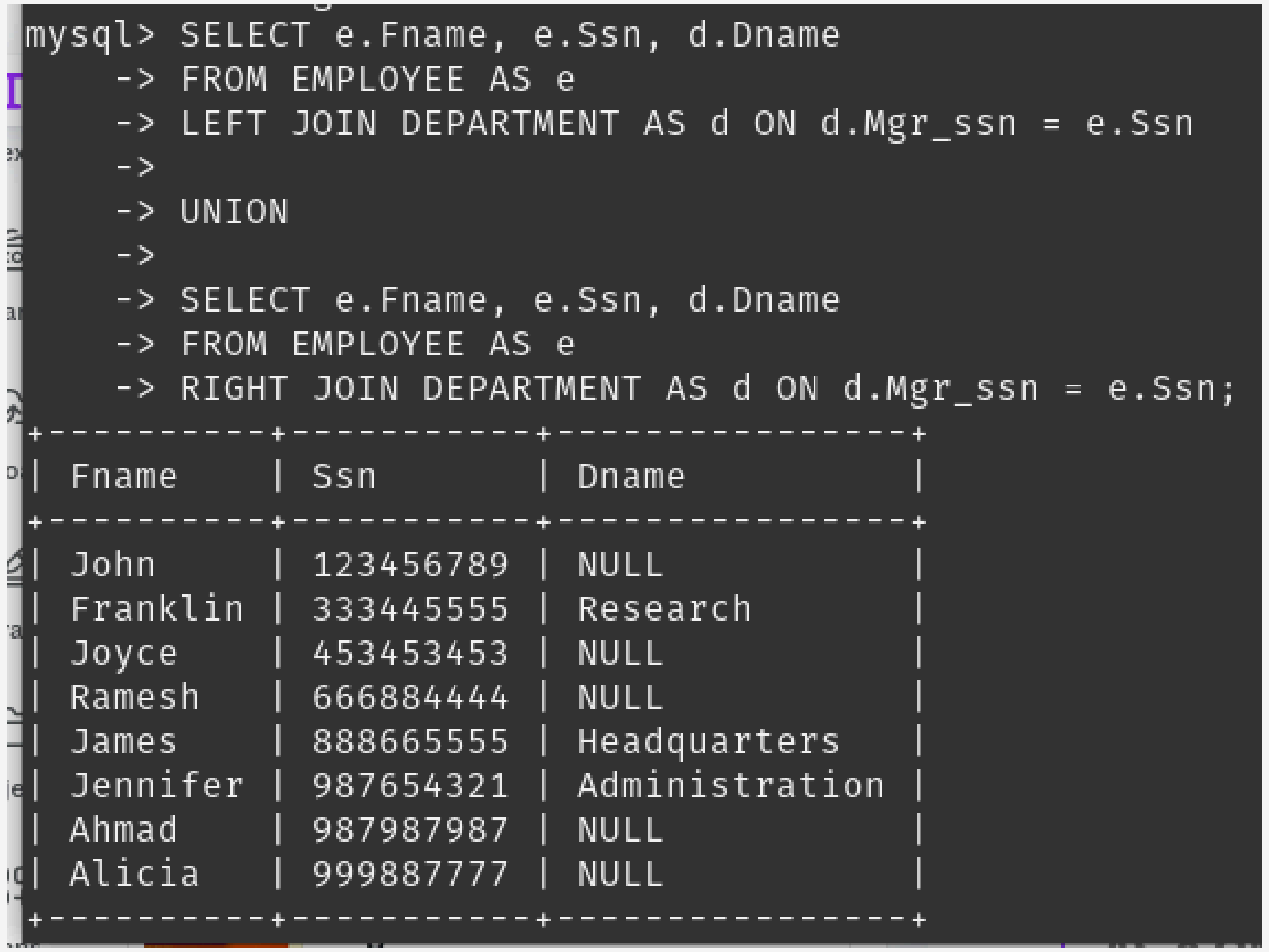
```
mysql> SELECT e.Fname, e.Ssn , DEPARTMENT.Dname FROM DEPARTMENT RIGHT JOIN EMPLO
YEE AS e ON DEPARTMENT.Mgr_ssn = e.Ssn;
+----------+-----------+----------------+
| Fname    | Ssn       | Dname          |
+----------+-----------+----------------+
| John     | 123456789 | NULL           |
| Franklin | 333445555 | Research       |
| Joyce    | 453453453 | NULL           |
| Ramesh   | 666884444 | NULL           |
| James    | 888665555 | Headquarters   |
| Jennifer | 987654321 | Administration |
| Ahmad    | 987987987 | NULL           |
| Alicia   | 999887777 | NULL           |
+----------+-----------+----------------+
8 rows in set (0.00 sec)
```

FULL OUTER JOIN OR FULL JOIN

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

FULL OUTER JOIN can potentially return very large result-sets!

MySQL does not directly support FULL JOIN. However, you can achieve the same result by combining LEFT JOIN and RIGHT JOIN using UNION.



```
mysql> SELECT e.Fname, e.Ssn, d.Dname
    -> FROM EMPLOYEE AS e
    -> LEFT JOIN DEPARTMENT AS d ON d.Mgr_ssn = e.Ssn
    ->
    -> UNION
    ->
    -> SELECT e.Fname, e.Ssn, d.Dname
    -> FROM EMPLOYEE AS e
    -> RIGHT JOIN DEPARTMENT AS d ON d.Mgr_ssn = e.Ssn;
+----------+-----------+----------------+
| Fname    | Ssn       | Dname          |
+----------+-----------+----------------+
| John     | 123456789 | NULL           |
| Franklin | 333445555 | Research       |
| Joyce    | 453453453 | NULL           |
| Ramesh   | 666884444 | NULL           |
| James    | 888665555 | Headquarters   |
| Jennifer | 987654321 | Administration |
| Ahmad    | 987987987 | NULL           |
| Alicia   | 999887777 | NULL           |
+----------+-----------+----------------+
```

Some DATE-TIME queries

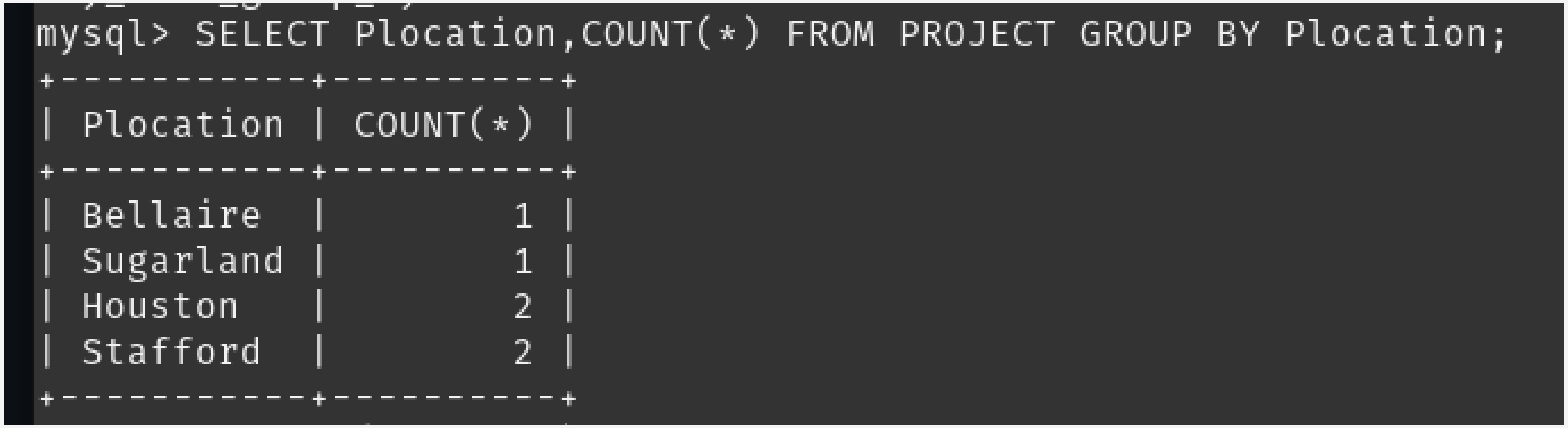
```
SELECT CURDATE();  or SELECT CURRENT_DATE();

SELECT CURRENT_TIME();  or SELECT CURTIME();

SELECT DATE_FORMAT("2017-06-15", "%Y");

SELECT DATE_SUB("2017-06-15", INTERVAL 10 DAY);

SELECT ADDTIME("2017-06-15 09:34:21.000001", "5.000003");
```

Add 5 seconds and 3 microseconds to a time and return the datetime https://www.w3schools.com/sql/

GROUP BY

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

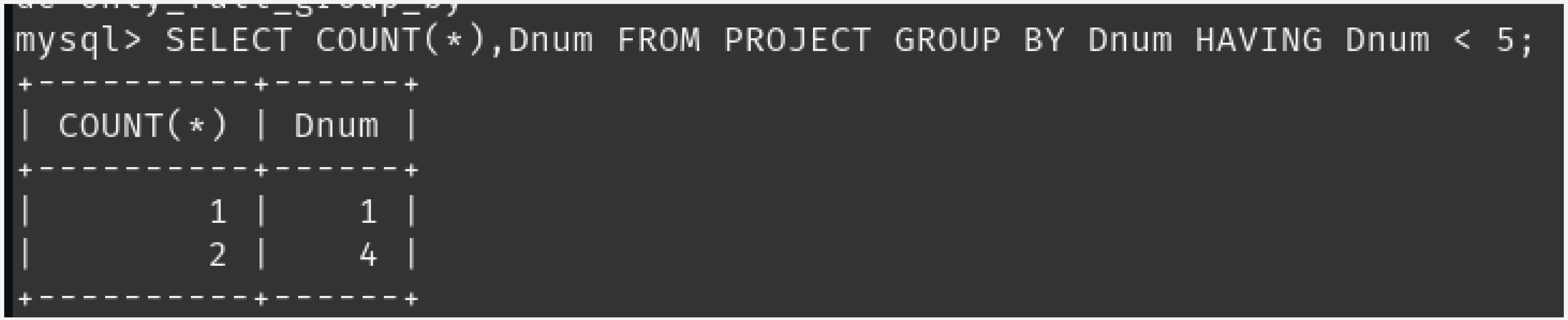
The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.



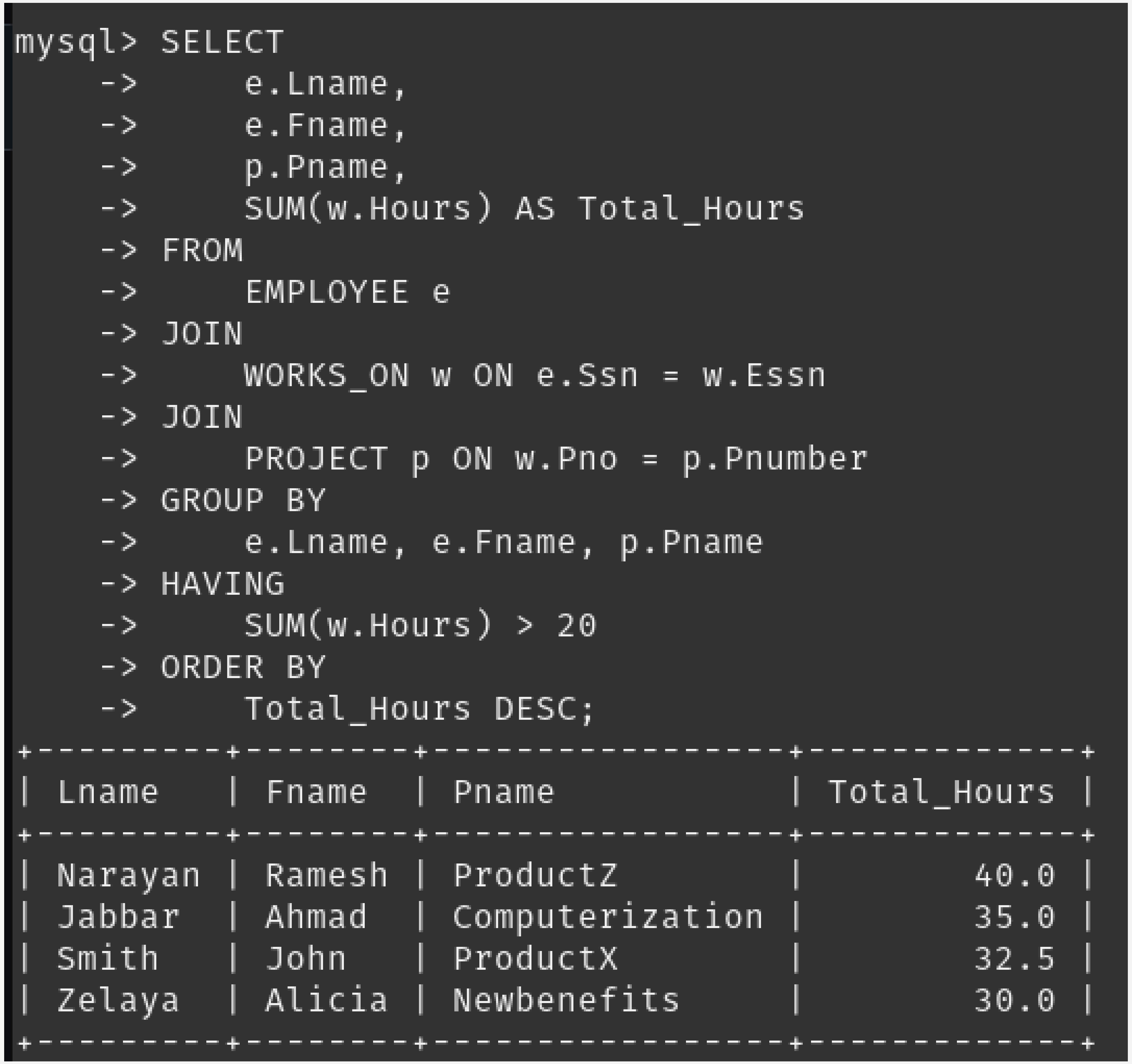
```
mysql> SELECT Plocation,COUNT(*) FROM PROJECT GROUP BY Plocation;
+-----------+----------+
| Plocation | COUNT(*) |
+-----------+----------+
| Bellaire  |        1 |
| Sugarland |        1 |
| Houston   |        2 |
| Stafford  |        2 |
+-----------+----------+
```

HAVING

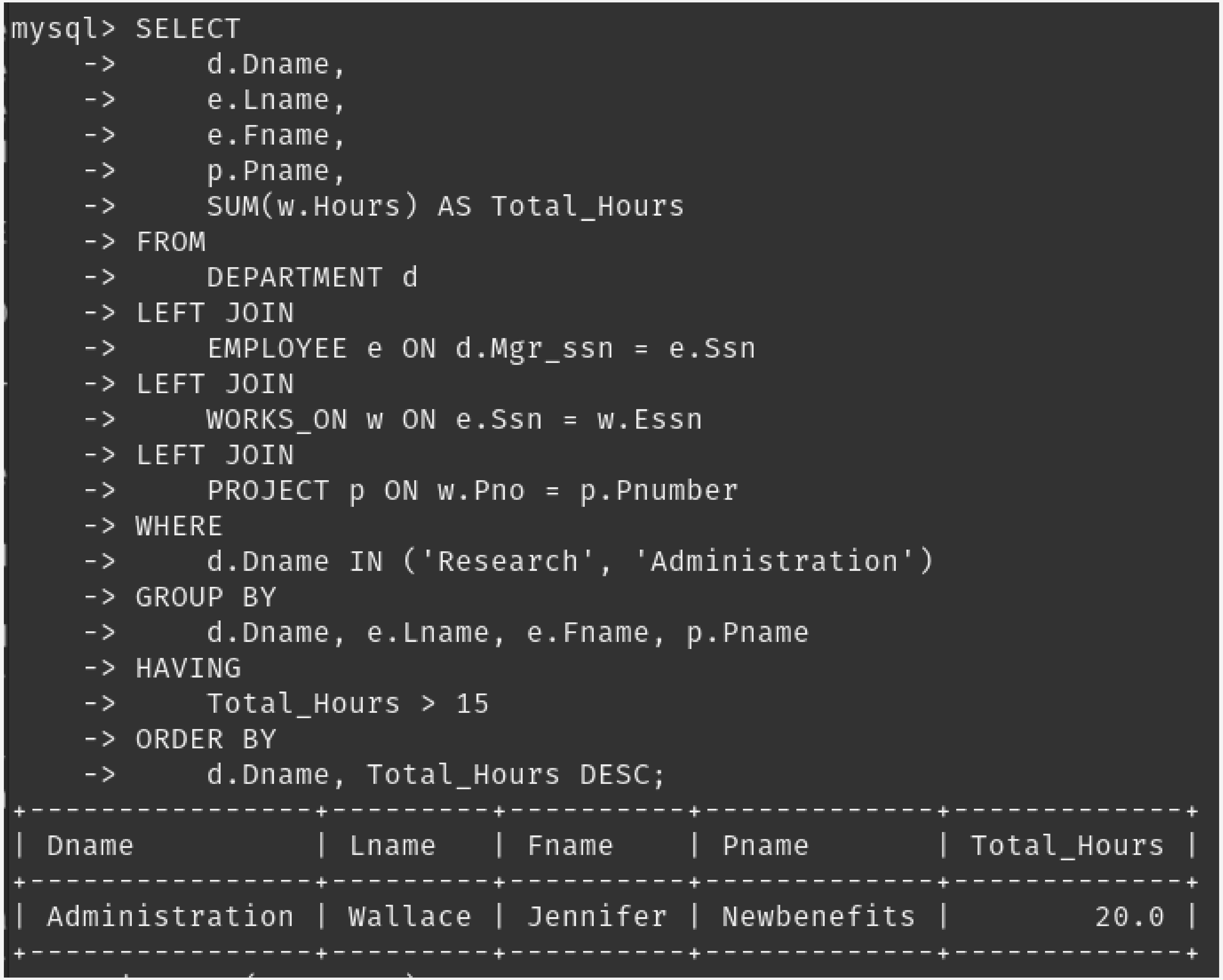
The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.



```
mysql> SELECT COUNT(*),Dnum FROM PROJECT GROUP BY Dnum HAVING Dnum < 5;
+----------+------+
| COUNT(*) | Dnum |
+----------+------+
|        1 |    1 |
|        2 |    4 |
+----------+------+
```



```
mysql> SELECT
    ->     e.Lname,
    ->     e.Fname,
    ->     p.Pname,
    ->     SUM(w.Hours) AS Total_Hours
    -> FROM
    ->     EMPLOYEE e
    -> JOIN
    ->     WORKS_ON w ON e.Ssn = w.Essn
    -> JOIN
    ->     PROJECT p ON w.Pno = p.Pnumber
    -> GROUP BY
    ->     e.Lname, e.Fname, p.Pname
    -> HAVING
    ->     SUM(w.Hours) > 20
    -> ORDER BY
    ->     Total_Hours DESC;
+---------+--------+-----------------+-------------+
| Lname   | Fname  | Pname           | Total_Hours |
+---------+--------+-----------------+-------------+
| Narayan | Ramesh | ProductZ        |        40.0 |
| Jabbar  | Ahmad  | Computerization |        35.0 |
| Smith   | John   | ProductX        |        32.5 |
| Zelaya  | Alicia | Newbenefits     |        30.0 |
+---------+--------+-----------------+-------------+
```



```
mysql> SELECT
    ->     d.Dname,
    ->     e.Lname,
    ->     e.Fname,
    ->     p.Pname,
    ->     SUM(w.Hours) AS Total_Hours
    -> FROM
    ->     DEPARTMENT d
    -> LEFT JOIN
    ->     EMPLOYEE e ON d.Mgr_ssn = e.Ssn
    -> LEFT JOIN
    ->     WORKS_ON w ON e.Ssn = w.Essn
    -> LEFT JOIN
    ->     PROJECT p ON w.Pno = p.Pnumber
    -> WHERE
    ->     d.Dname IN ('Research', 'Administration')
    -> GROUP BY
    ->     d.Dname, e.Lname, e.Fname, p.Pname
    -> HAVING
    ->     Total_Hours > 15
    -> ORDER BY
    ->     d.Dname, Total_Hours DESC;
+----------------+---------+----------+-------------+-------------+
| Dname          | Lname   | Fname    | Pname       | Total_Hours |
+----------------+---------+----------+-------------+-------------+
| Administration | Wallace | Jennifer | Newbenefits |        20.0 |
+----------------+---------+----------+-------------+-------------+
```