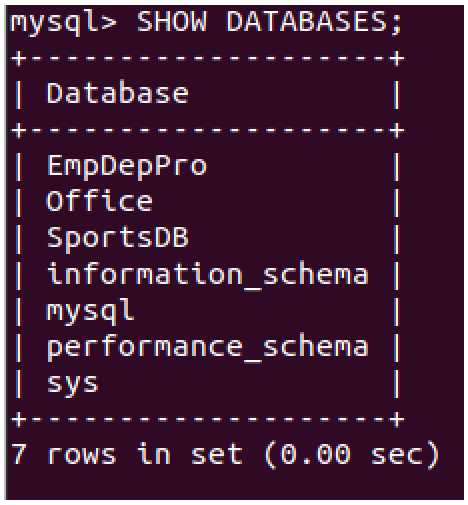
CS23I1014 – Bhadresh L
DBS – Lab 5 – Nested Queries and Aggregates

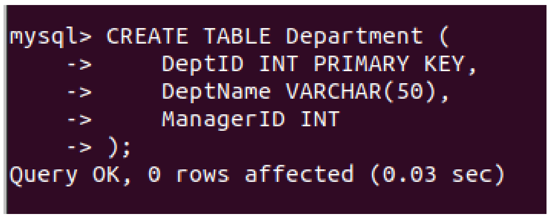
Tables Creation

Office Database Created:

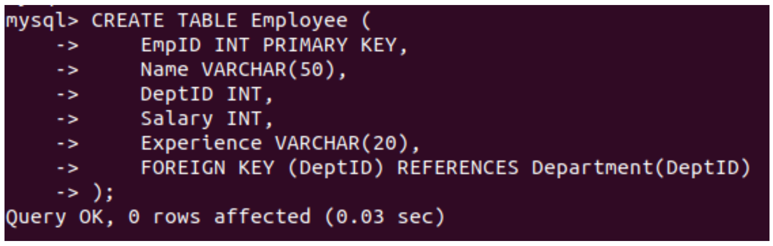


```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| EmpDepPro          |
| Office             |
| SportsDB           |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
7 rows in set (0.00 sec)
```

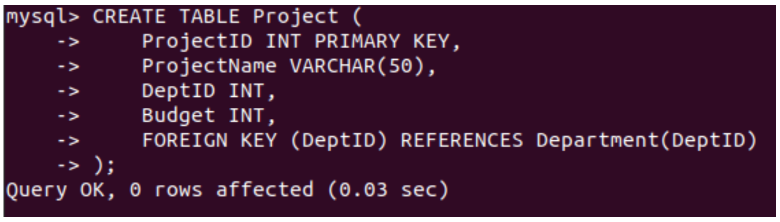
Tables Created:



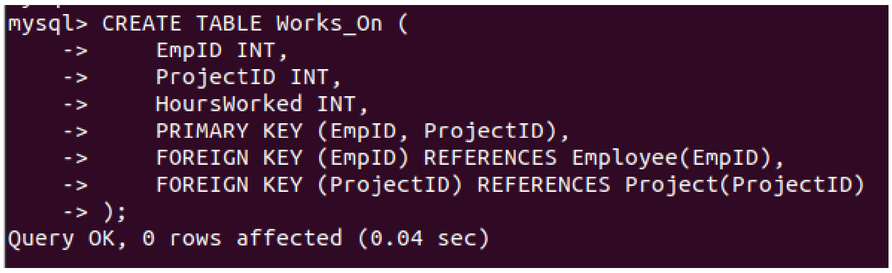
```
mysql> CREATE TABLE Department (
    ->     DeptID INT PRIMARY KEY,
    ->     DeptName VARCHAR(50),
    ->     ManagerID INT
    -> );
Query OK, 0 rows affected (0.03 sec)
```



```
mysql> CREATE TABLE Employee (
    ->     EmpID INT PRIMARY KEY,
    ->     Name VARCHAR(50),
    ->     DeptID INT,
    ->     Salary INT,
    ->     Experience VARCHAR(20),
    ->     FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
    -> );
Query OK, 0 rows affected (0.03 sec)
```



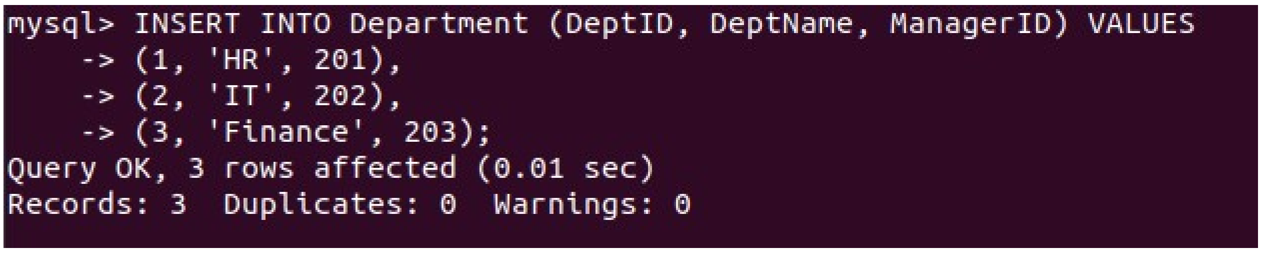
```
mysql> CREATE TABLE Project (
    ->     ProjectID INT PRIMARY KEY,
    ->     ProjectName VARCHAR(50),
    ->     DeptID INT,
    ->     Budget INT,
    ->     FOREIGN KEY (DeptID) REFERENCES Department(DeptID)
    -> );
Query OK, 0 rows affected (0.03 sec)
```



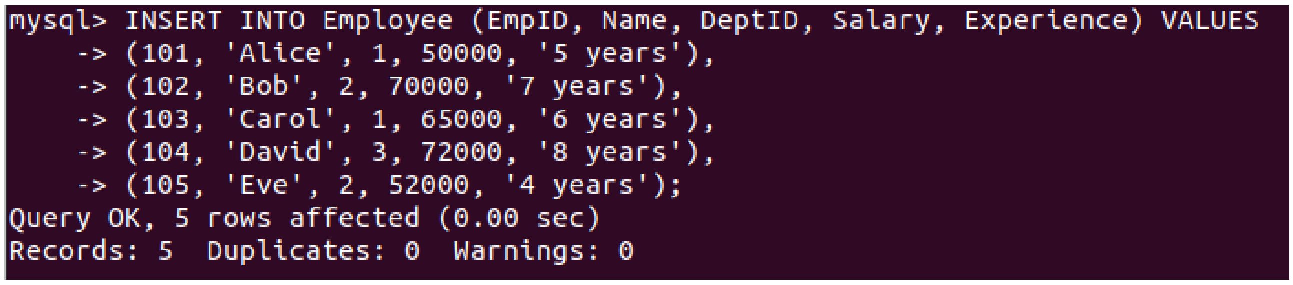
```
mysql> CREATE TABLE Works_On (
    ->     EmpID INT,
    ->     ProjectID INT,
    ->     HoursWorked INT,
    ->     PRIMARY KEY (EmpID, ProjectID),
    ->     FOREIGN KEY (EmpID) REFERENCES Employee(EmpID),
    ->     FOREIGN KEY (ProjectID) REFERENCES Project(ProjectID)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

DBS – Lab 5 – Nested Queries and Aggregates

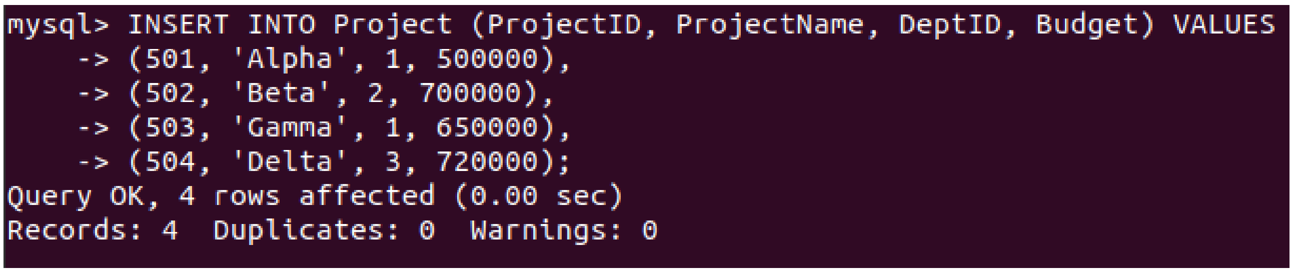
Tables Populated:



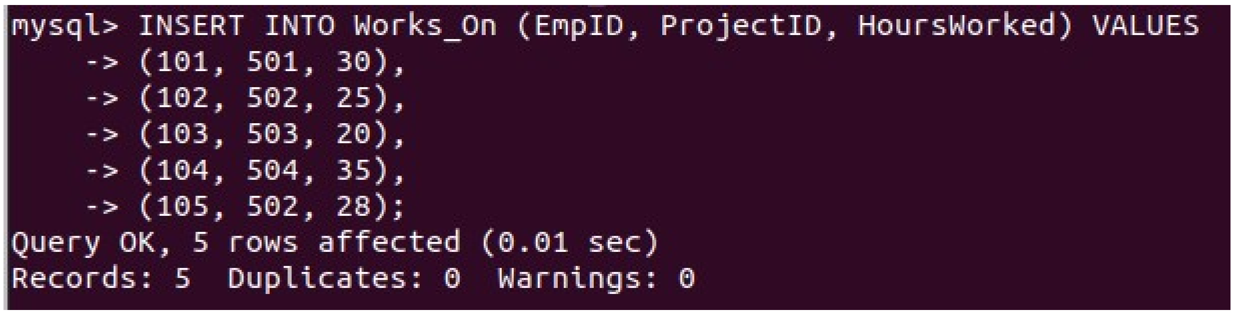
```
mysql> INSERT INTO Department (DeptID, DeptName, ManagerID) VALUES
    -> (1, 'HR', 201),
    -> (2, 'IT', 202),
    -> (3, 'Finance', 203);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```



```
mysql> INSERT INTO Employee (EmpID, Name, DeptID, Salary, Experience) VALUES
    -> (101, 'Alice', 1, 50000, '5 years'),
    -> (102, 'Bob', 2, 70000, '7 years'),
    -> (103, 'Carol', 1, 65000, '6 years'),
    -> (104, 'David', 3, 72000, '8 years'),
    -> (105, 'Eve', 2, 52000, '4 years');
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0
```



```
mysql> INSERT INTO Project (ProjectID, ProjectName, DeptID, Budget) VALUES
    -> (501, 'Alpha', 1, 500000),
    -> (502, 'Beta', 2, 700000),
    -> (503, 'Gamma', 1, 650000),
    -> (504, 'Delta', 3, 720000);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

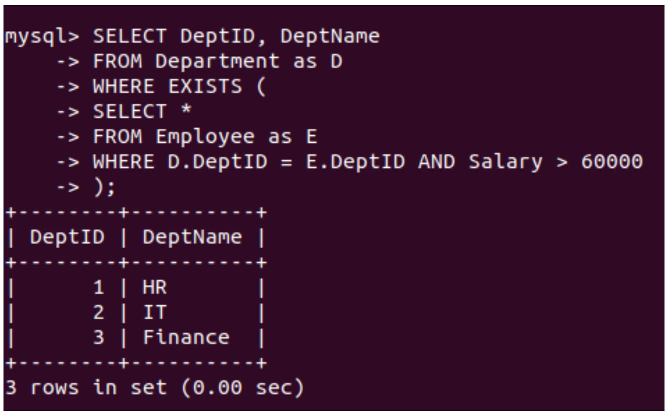


```
mysql> INSERT INTO Works_On (EmpID, ProjectID, HoursWorked) VALUES
    -> (101, 501, 30),
    -> (102, 502, 25),
    -> (103, 503, 20),
    -> (104, 504, 35),
    -> (105, 502, 28);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

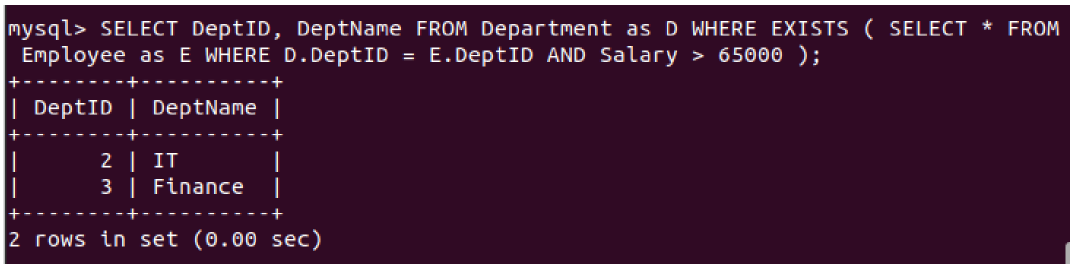
CS23I1014 – Bhadresh L
DBS – Lab 5 – Nested Queries and Aggregates

Nested Queries

1. Find departments that have at least one employee with a salary greater than 60,000 (I have shown 65000 also as there exists at least 1 employee in each department with greater than 60,000)**:**

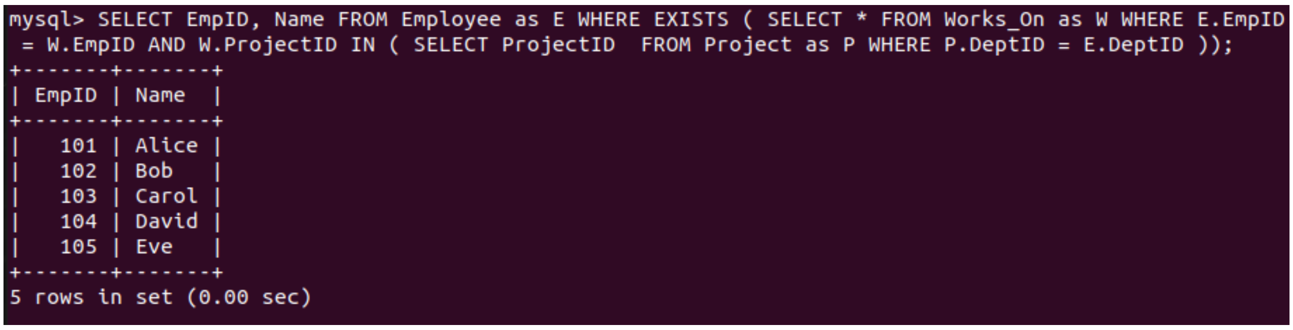


```
mysql> SELECT DeptID, DeptName
    -> FROM Department as D
    -> WHERE EXISTS (
    -> SELECT *
    -> FROM Employee as E
    -> WHERE D.DeptID = E.DeptID AND Salary > 60000
    -> );
+--------+----------+
| DeptID | DeptName |
+--------+----------+
|      1 | HR       |
|      2 | IT       |
|      3 | Finance  |
+--------+----------+
3 rows in set (0.00 sec)
```



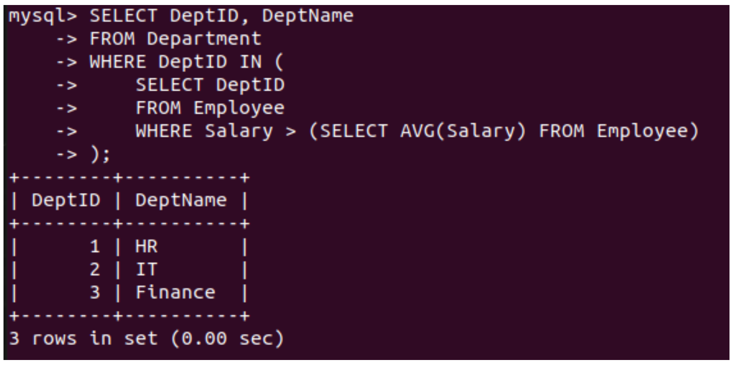
```
mysql> SELECT DeptID, DeptName FROM Department as D WHERE EXISTS ( SELECT * FROM
 Employee as E WHERE D.DeptID = E.DeptID AND Salary > 65000 );
+--------+----------+
| DeptID | DeptName |
+--------+----------+
|      2 | IT       |
|      3 | Finance  |
+--------+----------+
2 rows in set (0.00 sec)
```

2. Find employees who work on projects in their own department:



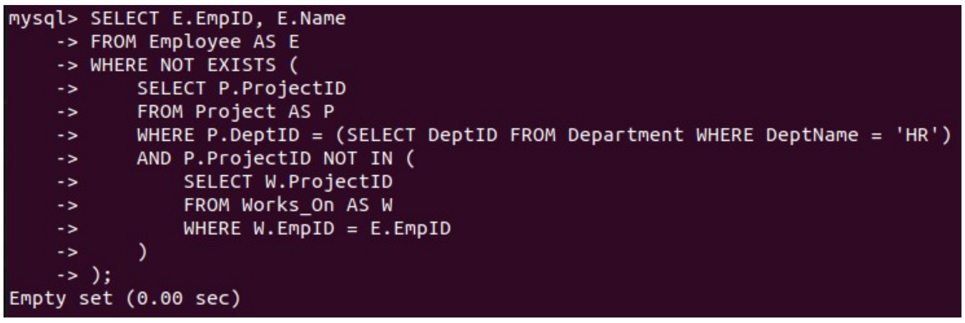
```
mysql> SELECT EmpID, Name FROM Employee as E WHERE EXISTS ( SELECT * FROM Works_On as W WHERE E.EmpID
 = W.EmpID AND W.ProjectID IN ( SELECT ProjectID  FROM Project as P WHERE P.DeptID = E.DeptID ));
+-------+-------+
| EmpID | Name  |
+-------+-------+
|   101 | Alice |
|   102 | Bob   |
|   103 | Carol |
|   104 | David |
|   105 | Eve   |
+-------+-------+
5 rows in set (0.00 sec)
```

3. Find departments where all employees earn more than the average salary of the entire company:



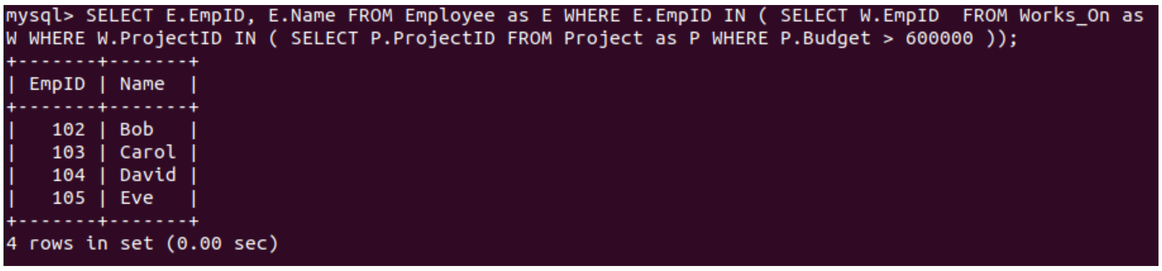
```
mysql> SELECT DeptID, DeptName
    -> FROM Department
    -> WHERE DeptID IN (
    ->     SELECT DeptID
    ->     FROM Employee
    ->     WHERE Salary > (SELECT AVG(Salary) FROM Employee)
    -> );
+--------+----------+
| DeptID | DeptName |
+--------+----------+
|      1 | HR       |
|      2 | IT       |
|      3 | Finance  |
+--------+----------+
3 rows in set (0.00 sec)
```

4. Find employees who work on all projects handled by the 'HR' department:



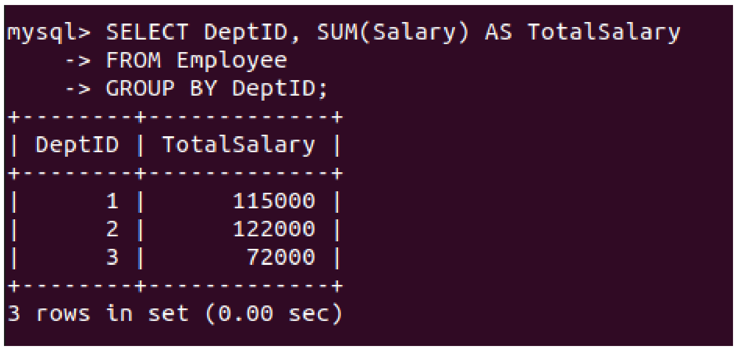
```
mysql> SELECT E.EmpID, E.Name
    -> FROM Employee AS E
    -> WHERE NOT EXISTS (
    ->     SELECT P.ProjectID
    ->     FROM Project AS P
    ->     WHERE P.DeptID = (SELECT DeptID FROM Department WHERE DeptName = 'HR')
    ->     AND P.ProjectID NOT IN (
    ->         SELECT W.ProjectID
    ->         FROM Works_On AS W
    ->         WHERE W.EmpID = E.EmpID
    ->     )
    -> );
Empty set (0.00 sec)
```

5. Find employees who are working on projects with a budget greater than 600,000:



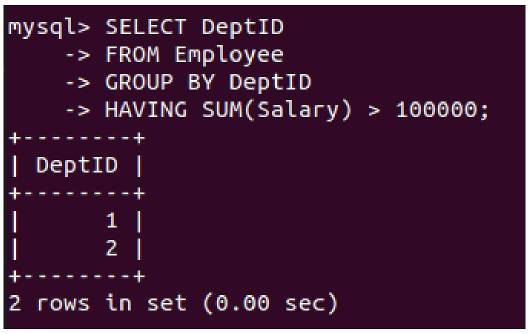
```
mysql> SELECT E.EmpID, E.Name FROM Employee as E WHERE E.EmpID IN ( SELECT W.EmpID  FROM Works_On as
W WHERE W.ProjectID IN ( SELECT P.ProjectID FROM Project as P WHERE P.Budget > 600000 ));
+-------+-------+
| EmpID | Name  |
+-------+-------+
|   102 | Bob   |
|   103 | Carol |
|   104 | David |
|   105 | Eve   |
+-------+-------+
4 rows in set (0.00 sec)
```

6. Find the total salary paid in each department:



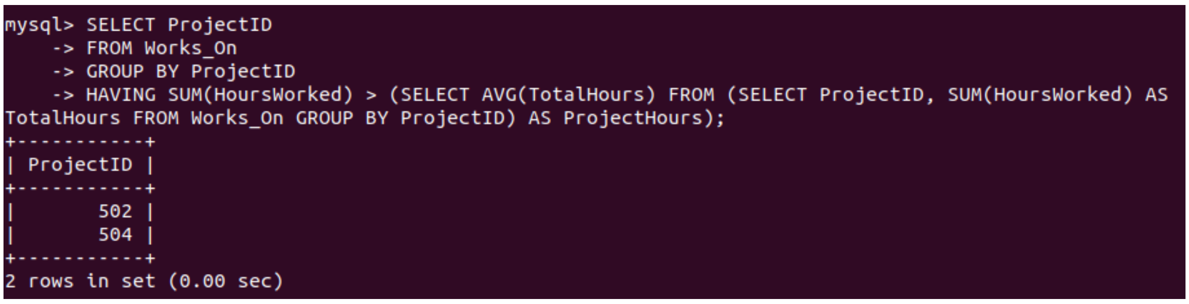
```
mysql> SELECT DeptID, SUM(Salary) AS TotalSalary
    -> FROM Employee
    -> GROUP BY DeptID;
+--------+-------------+
| DeptID | TotalSalary |
+--------+-------------+
|      1 |      115000 |
|      2 |      122000 |
|      3 |       72000 |
+--------+-------------+
3 rows in set (0.00 sec)
```

7. Find departments where the total salary of employees exceeds 100,000:



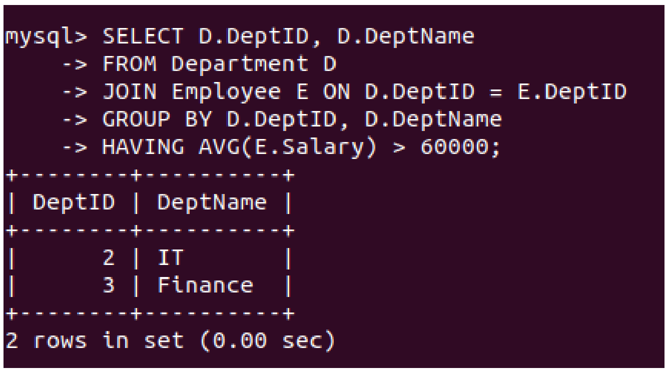
```
mysql> SELECT DeptID
    -> FROM Employee
    -> GROUP BY DeptID
    -> HAVING SUM(Salary) > 100000;
+--------+
| DeptID |
+--------+
|      1 |
|      2 |
+--------+
2 rows in set (0.00 sec)
```

8. Find projects where the total hours worked exceeds the average hours worked across all projects:



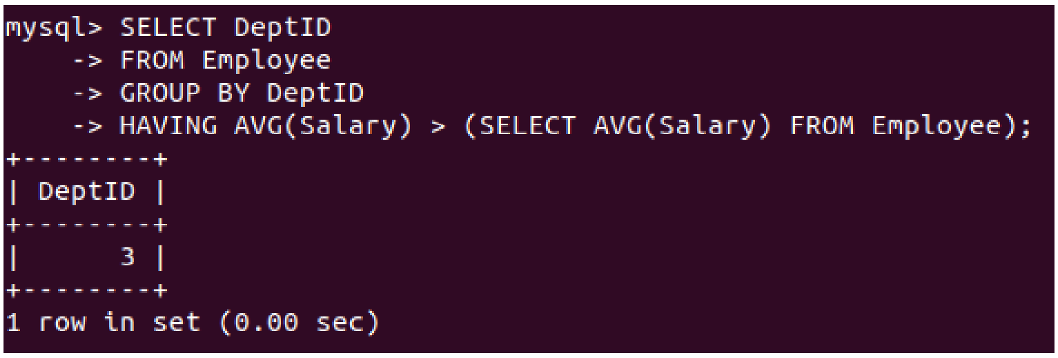
```
mysql> SELECT ProjectID
    -> FROM Works_On
    -> GROUP BY ProjectID
    -> HAVING SUM(HoursWorked) > (SELECT AVG(TotalHours) FROM (SELECT ProjectID, SUM(HoursWorked) AS
TotalHours FROM Works_On GROUP BY ProjectID) AS ProjectHours);
+-----------+
| ProjectID |
+-----------+
|       502 |
|       504 |
+-----------+
2 rows in set (0.00 sec)
```

9. Find departments where the average salary of employees is greater than 60,000 (Using Join):



```
mysql> SELECT D.DeptID, D.DeptName
    -> FROM Department D
    -> JOIN Employee E ON D.DeptID = E.DeptID
    -> GROUP BY D.DeptID, D.DeptName
    -> HAVING AVG(E.Salary) > 60000;
+--------+----------+
| DeptID | DeptName |
+--------+----------+
|      2 | IT       |
|      3 | Finance  |
+--------+----------+
2 rows in set (0.00 sec)
```

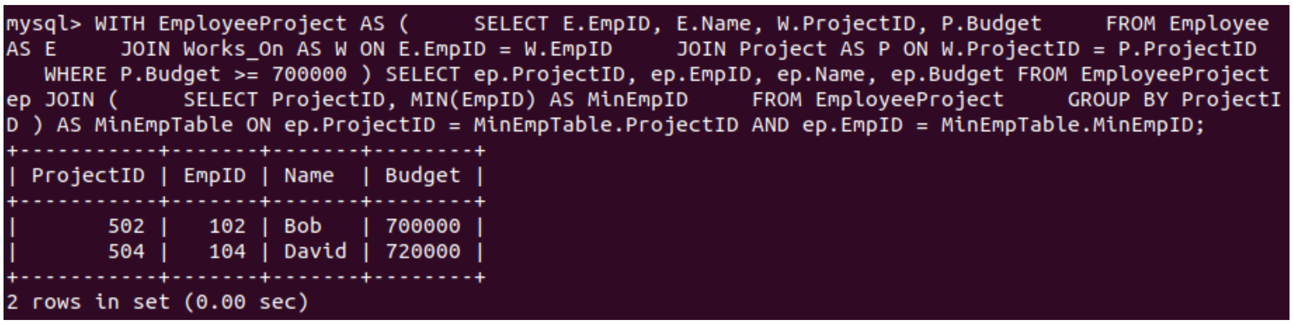
10. Find departments where the average salary is higher than the overall company's average salary (Using a nested subquery in the Having Clause):



```
mysql> SELECT DeptID
    -> FROM Employee
    -> GROUP BY DeptID
    -> HAVING AVG(Salary) > (SELECT AVG(Salary) FROM Employee);
+--------+
| DeptID |
+--------+
|      3 |
+--------+
1 row in set (0.00 sec)
```

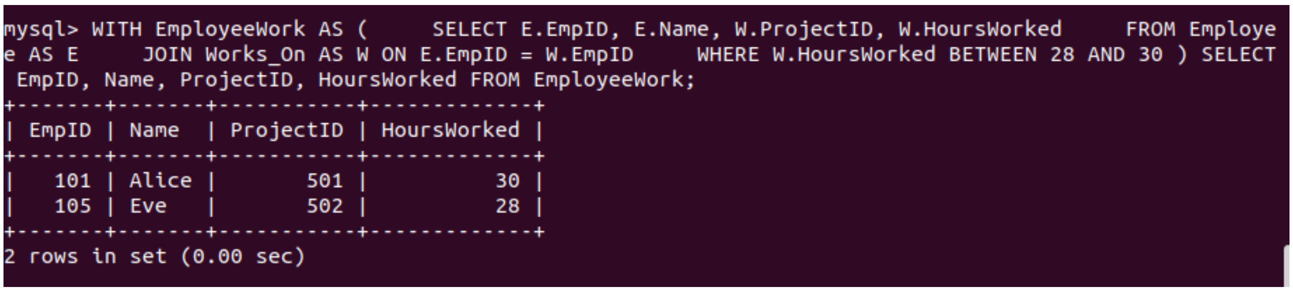
DBS – Lab 5 – Nested Queries and Aggregates

11. By observation, we can see that the query filters the employees working on projects having budget greater than or equal to 700000. It returns EmpID, Name, ProjectID and Budget. We can Join the 2 tables, Employee and Projects, on EmpID and Filter with respect to budget, projecting only the returned columns. The query and its result are as follows:



```
mysql> WITH EmployeeProject AS (     SELECT E.EmpID, E.Name, W.ProjectID, P.Budget     FROM Employee
AS E     JOIN Works_On AS W ON E.EmpID = W.EmpID     JOIN Project AS P ON W.ProjectID = P.ProjectID
   WHERE P.Budget >= 700000 ) SELECT ep.ProjectID, ep.EmpID, ep.Name, ep.Budget FROM EmployeeProject
ep JOIN (     SELECT ProjectID, MIN(EmpID) AS MinEmpID     FROM EmployeeProject     GROUP BY ProjectI
D ) AS MinEmpTable ON ep.ProjectID = MinEmpTable.ProjectID AND ep.EmpID = MinEmpTable.MinEmpID;
+-----------+-------+-------+--------+
| ProjectID | EmpID | Name  | Budget |
+-----------+-------+-------+--------+
|       502 |   102 | Bob   | 700000 |
|       504 |   104 | David | 720000 |
+-----------+-------+-------+--------+
2 rows in set (0.00 sec)
```

12. By observation, we can see that the number of hours worked lies between 28 and 30, hence that shall be the filter. We can join the Employee and Project table for the same, apply the condition and only filter the required columns. The query and its result are as follows:



```
mysql> WITH EmployeeWork AS (     SELECT E.EmpID, E.Name, W.ProjectID, W.HoursWorked     FROM Employe
e AS E     JOIN Works_On AS W ON E.EmpID = W.EmpID     WHERE W.HoursWorked BETWEEN 28 AND 30 ) SELECT
 EmpID, Name, ProjectID, HoursWorked FROM EmployeeWork;
+-------+-------+-----------+-------------+
| EmpID | Name  | ProjectID | HoursWorked |
+-------+-------+-----------+-------------+
|   101 | Alice |       501 |          30 |
|   105 | Eve   |       502 |          28 |
+-------+-------+-----------+-------------+
2 rows in set (0.00 sec)
```