LECTURE 5: STRUCTURE QUERY LANGUAGE (SQL) 3

COMP2004J: Databases and Information Systems

Dr. Ruihai Dong (ruihai.dong@ucd.ie)
UCD School of Computer Science
Beijing-Dublin International College

Today’s Outline

- AUTO_INCREMENT fields.
- Ordering Data (ORDER BY)
- Limiting the number of rows (LIMIT)
- Aggregate Queries (COUNT, MAX, MIN, SUM, AVG)
- Grouping Data (GROUP BY)
- Nested Queries
- Views

Tables

employee

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60

department

deptno	deptname	office	division	managernumber
10	Training	Lansdown	D1	4567
20	Design	Belfield	D2	1899
30	Implementation	Donnybrook	D1	1345
40	Finance	Ballsbridge	D3	1595

I have uploaded these tables to Moodle as **“employee.db”** so you can try these queries yourself.

AUTO INCREMENT

AUTO_INCREMENT

- If you use an INT value as a primary key, you can ask the RDBMS to automatically generate the values for you using **AUTO_INCREMENT**.
- The values begin at 1 and increase by 1 each time you insert a new row.
- Put a NULL value into the AUTO_INCREMENT field when inserting a row.

AUTO_INCREMENT Example

- `CREATE TABLE test (id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(30));`
- Two ways to insert:
  1. Don't provide a value for the AUTO_INCREMENT field, e.g. `INSERT INTO test (name) VALUES ('John');`

  2. Insert a NULL value into the AUTO_INCREMENT field, e.g. `INSERT INTO test VALUES( NULL, 'Barry');`

```
mysql> SELECT * FROM test;
+----+-------+
| id | name  |
+----+-------+
|  1 | John  |
|  2 | Barry |
+----+-------+
2 rows in set (0.00 sec)
```

ORDERING DATA

ORDER BY

- The ORDER BY clause, at the end of the query, orders the rows of the result.

- Example:
- `SELECT * FROM employee ORDER BY lastname, firstname;`
  - Employees are ordered by `lastname`.
  - If two employees have the same last name, they will be ordered by their `firstname` attributes.
  - The default behaviour is to sort in **ascending** order (i.e. from A to Z).

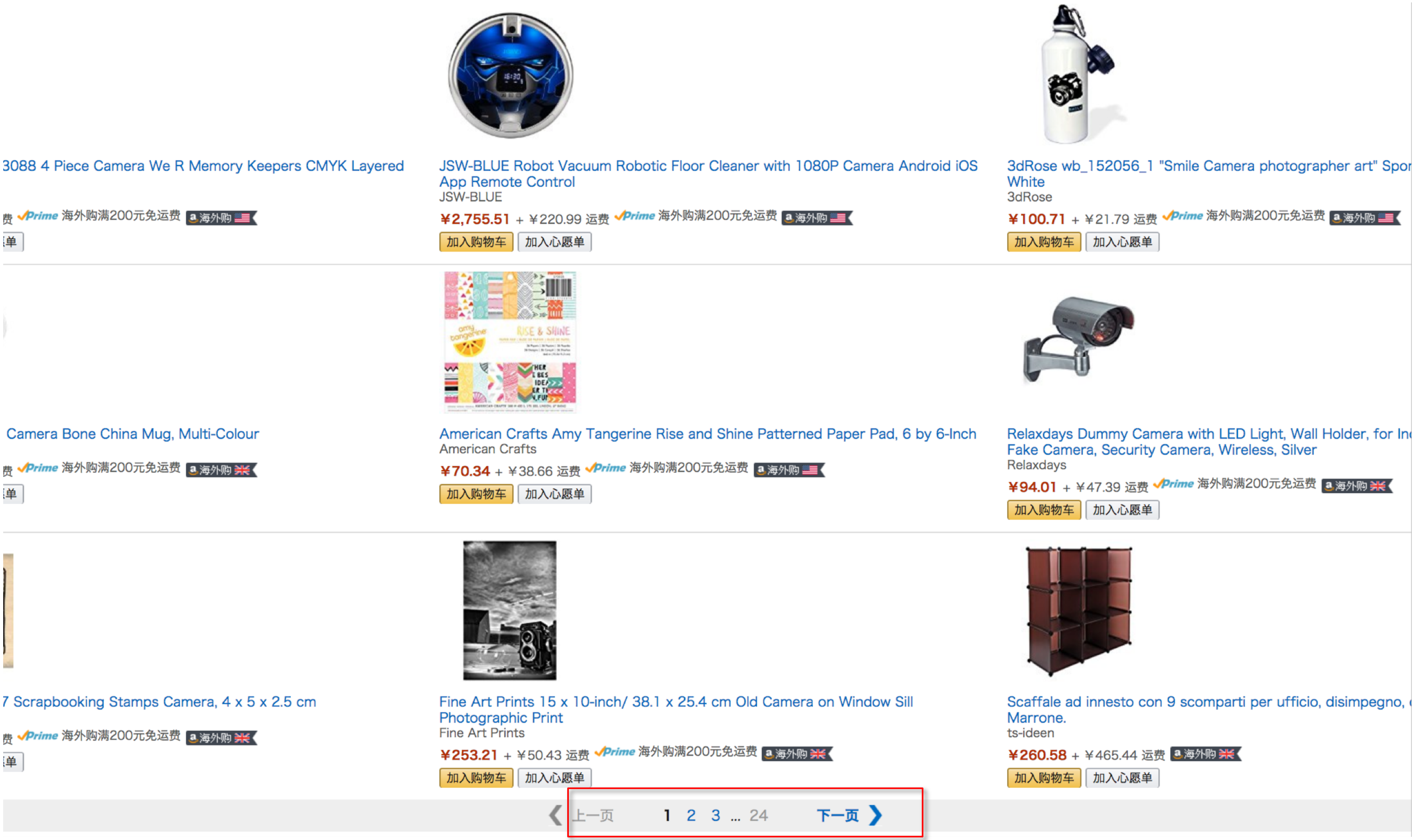
ORDER BY

- Example:

- `SELECT * FROM employee ORDER BY salary DESC;`
  - Employees are ordered by their `salary`.
  - Here, we have specified that they are to be ordered in **descending order** (DESC), from the highest salary to the lowest.

LIMITING THE NUMBER OF ROWS

Limiting the number of rows



3088 4 Piece Camera We R Memory Keepers CMYK Layered

费 ✓Prime 海外购满200元免运费 a 海外购

单

JSW-BLUE Robot Vacuum Robotic Floor Cleaner with 1080P Camera Android iOS App Remote Control
JSW-BLUE
¥2,755.51 + ¥220.99 运费 ✓Prime 海外购满200元免运费 a 海外购

加入购物车 加入心愿单

3dRose wb_152056_1 "Smile Camera photographer art" Spor White
3dRose
¥100.71 + ¥21.79 运费 ✓Prime 海外购满200元免运费 a 海外购

加入购物车 加入心愿单

Camera Bone China Mug, Multi-Colour

费 ✓Prime 海外购满200元免运费 a 海外购

单

American Crafts Amy Tangerine Rise and Shine Patterned Paper Pad, 6 by 6-Inch
American Crafts
¥70.34 + ¥38.66 运费 ✓Prime 海外购满200元免运费 a 海外购

加入购物车 加入心愿单

Relaxdays Dummy Camera with LED Light, Wall Holder, for In Fake Camera, Security Camera, Wireless, Silver
Relaxdays
¥94.01 + ¥47.39 运费 ✓Prime 海外购满200元免运费 a 海外购

加入购物车 加入心愿单

7 Scrapbooking Stamps Camera, 4 x 5 x 2.5 cm

费 ✓Prime 海外购满200元免运费 a 海外购

单

Fine Art Prints 15 x 10-inch/ 38.1 x 25.4 cm Old Camera on Window Sill Photographic Print
Fine Art Prints
¥253.21 + ¥50.43 运费 ✓Prime 海外购满200元免运费 a 海外购

加入购物车 加入心愿单

Scaffale ad innesto con 9 scomparti per ufficio, disimpegno, Marrone.
ts-ideen
¥260.58 + ¥465.44 运费 a 海外购

加入购物车 加入心愿单

❮ 上一页 1 2 3 ... 24 下一页 ❯

Limiting the number of rows

- We can use the LIMIT keyword to reduce the number of rows returned by a SELECT query.

- To show only 10 employees' data:
  - `SELECT * FROM employee LIMIT 10;`

- Find the name of the employee with the highest salary:

  -
    ```
    SELECT firstname, familyname FROM employee
      ORDER BY salary DESC LIMIT 1;
    ```

Limiting the number of rows

- To show only top 10 employees’ data:
  - `SELECT * FROM employee LIMIT 10;`
- To show 11-20 employees’ data:
  - `SELECT * FROM employee LIMIT 10,10;`

AGGREGATE QUERIES

Aggregate queries

- The result of an aggregate query depends on **sets of rows**.

- SQL-2 offers five aggregate operators:
    - `COUNT`
    - `SUM`
    - `MAX`
    - `MIN`
    - `AVG`

COUNT (count values/rows)

- COUNT returns the **number** of rows or distinct values;

- Examples
  - Find the number of employees (number of rows/tuples in the table):

    ```
    SELECT COUNT(*) FROM employee;
    ```

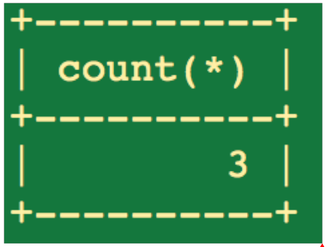
  - Find the number of values in a particular column (NULL values are not counted):
    - `SELECT COUNT(salary) FROM employee;`

  - Find the number of different values on the attribute salary for all the rows in the employee table:

    ```
    SELECT COUNT(DISTINCT salary) FROM employee;
    ```

COUNT (count values/rows)

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60



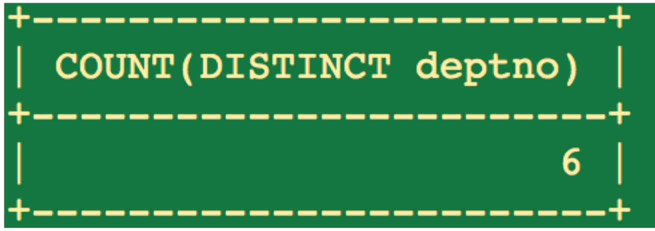
```
+----------+
| count(*) |
+----------+
|        3 |
+----------+
```

- Examples
  - Find the number of employees (number of rows/tuples in the table) in department 10:

```
SELECT COUNT(*) FROM employee where deptno=10;
```

COUNT (count values/rows)

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60



```
+------------------------+
| COUNT(DISTINCT deptno) |
+------------------------+
|                      6 |
+------------------------+
```

- Examples
  - Find the number of different values on the attribute deptno for all the rows in the employee table:

```
SELECT COUNT(DISTINCT deptno) FROM employee;
```

MAX and MIN

- The MAX function returns the **maximum** value in a set of values.
- The MIN function returns the **minimum** value in a set of values.

- Example
  - Find the highest/lowest salary paid to any employee:

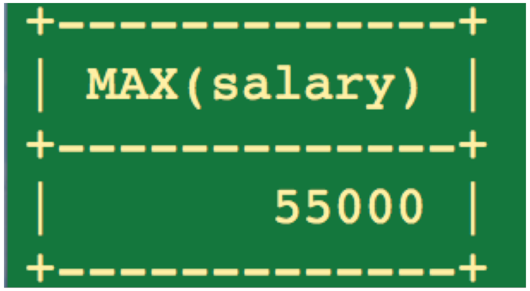
```
SELECT MAX(salary) FROM employee;
SELECT MIN(salary) FROM employee;
```

MAX and MIN

- Example
  - Find the highestsalary paid to any employee:

```
SELECT MAX(salary) FROM employee;
```

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60



```
+-------------+
| MAX(salary) |
+-------------+
|       55000 |
+-------------+
```

SUM and AVG

- The SUM function returns the **sum** of a set of value (ignores NULL values).
- The AVG function calculates the **average** value of a set of values (ignores NULL values).

- Example
  - Find the total salary bill for all employees:

```
SELECT SUM(salary) FROM employee;
```

  - Find the average salary of employees:

```
SELECT AVG(salary) FROM employee;
```

Aggregate queries with joins

- Find the highest paid employee who is a teacher.

```
SELECT MAX(salary) FROM employee
  WHERE job='Teacher’;
```

- Find the average salary of all employees in the design department

```
SELECT AVG(salary)
  FROM employee
  INNER JOIN department USING(deptno)
  WHERE deptname='Design';
```

Aggregate queries and target list

- Incorrect query:

```
SELECT firstname, familyname, MAX(salary)
  FROM employee;
```

- Attributes **must** have same number of results.

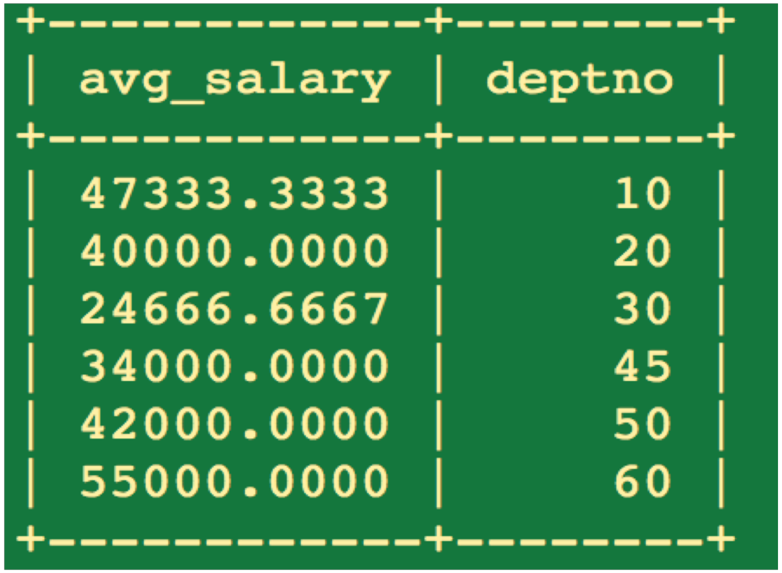
- Find the maximum and minimum salaries of all employees:

```
SELECT MAX(salary), MIN(salary) FROM employee;
```

GROUPING DATA

Grouping Data

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60



```
+------------+--------+
| avg_salary | deptno |
+------------+--------+
| 47333.3333 |     10 |
| 40000.0000 |     20 |
| 24666.6667 |     30 |
| 34000.0000 |     45 |
| 42000.0000 |     50 |
| 55000.0000 |     60 |
+------------+--------+
```

Grouping data (GROUP BY)

- Queries may apply aggregate operators to **subsets of rows**.

- Find the sum of salaries of all the employees of the same department:

```
SELECT deptname, SUM(salary) FROM employee
      INNER JOIN department USING(deptno)
      GROUP BY deptname;
```

GROUP BY – How it works

- First, the query is executed without GROUP BY and without aggregate operators.

- Then the query result is divided in subsets with the same values for the attributes appearing after the group by clause.

- Finally, the aggregate operator is applied separately to each subset.

Group predicates

- When conditions are on the result of an aggregate operator, it is necessary to use the HAVING clause.
- Find the departments where the average salary is over 40000.

```
SELECT deptno FROM employee GROUP BY deptno
     HAVING AVG(salary) > 40000;
```

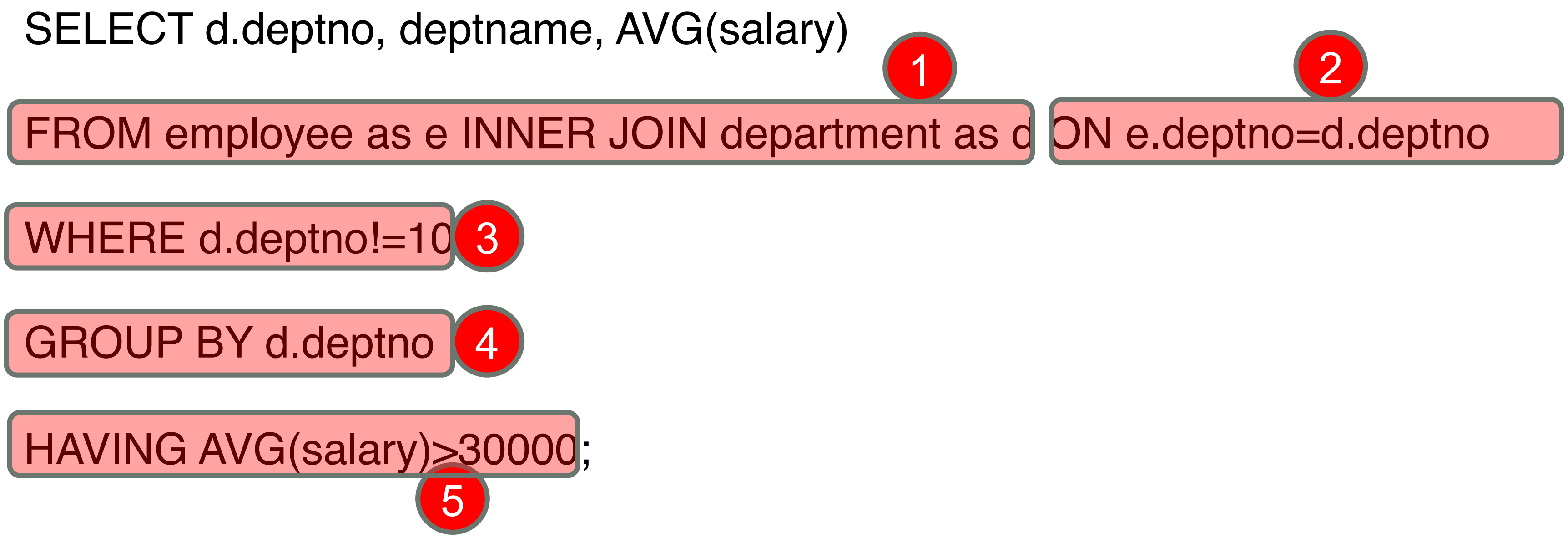
WHERE or HAVING?

- Only predicates containing aggregate operators should appear in the argument of the HAVING clause.

- Find the name of the departments where the average salary is over 40000.

```
SELECT deptname FROM employee
  INNER JOIN department USING(deptno)
  GROUP BY deptname HAVING AVG(salary) > 40000;
```

How it works? (Good example :D)



```
SELECT d.deptno, deptname, AVG(salary)
FROM employee as e INNER JOIN department as d ON e.deptno=d.deptno
WHERE d.deptno!=10
GROUP BY d.deptno
HAVING AVG(salary)>30000;
```

NESTED QUERIES

Nested Queries

- A **nested query** is an SQL query that is contained within another query.
  - The nested query is sometimes called a **subquery**.

- Most commonly used in the WHERE clause of a query.

- Subqueries can return:
  - A single value (known as a **scalar**)
  - A single column.
  - A single row.
  - A table (multiple columns/rows)

Scalars and single columns are the most common usages of subqueries.

Nested Queries: Scalar Value

- When a subquery returns a single value (scalar), it can be used in the same way as an ordinary single value.

- Find the name of the employee(s) who earn the most money:
  - SELECT firstname, familyname FROM employee WHERE salary=(SELECT MAX(salary) FROM employee);

Nested Queries: Single Columns

- A nested query returning a single column can be thought of as a **set of values**.

- We can compare attributes with values from this set to see if it's equal, greater than, less than, etc.
  - Using = > < >= <= <> !=

- Two other keywords are important:
  - ANY: returns true if the comparison is true for **any** value in the set.
  - ALL: returns true if the comparison is true for **all** the values in the set.

Nested Query: Single Columns Example 1

- Find the names of employees who work in departments in Division 'D1’.

```
SELECT firstname, familyname FROM employee
  WHERE deptno = ANY(SELECT deptno FROM
    department WHERE division='D1');
```

1. The subquery finds a list of all the department numbers (deptno) for departments that are in division D1.
2. When selecting from the ‘employee’ table, it will match any employee whose ‘deptno’ is in the set returned by the subquery.

Nested Query: Single Columns Example 1

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60

```
SELECT firstname,
familyname FROM employee
  WHERE deptno =
ANY(SELECT deptno FROM
    department WHERE
division='D1');
```

deptno	deptname	office	division	managernumber
10	Training	Lansdown	D1	4567
20	Design	Belfield	D2	1899
30	Implementation	Donnybrook	D1	1345
40	Finance	Ballsbridge	D3	1595

Nested Query: Single Columns Example 2

- Find the employees of department number 10 who have the same first name as a member of department 20.

```
SELECT firstname, familyname FROM employee
  WHERE deptno=10 AND firstname=ANY(
    SELECT firstname FROM employee WHERE
    deptno=20);
```

- (Of course, this particular requirement can also be achieved without a nested query.)

Nested Query: Single Columns Example 2

- Find the employees of department number 10 who have the same first name as a member of department 20.

```
SELECT firstname, familyname FROM employee
  WHERE deptno=10 AND firstname=ANY(
    SELECT firstname FROM employee WHERE
    deptno=20);
```

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60

Nested Query: Single Columns Example 3

- Find the name of the Department in which there is no employee named Sean.

```
SELECT deptname FROM department
   WHERE deptno != ALL(SELECT DISTINCT deptno
     FROM employee WHERE firstname='Sean');
```

- Note that we change from ANY to ALL because this is a negative match.
- The subquery finds the ‘deptno’ for departments who do have an employee named ‘Sean’.
- We want to find departments whose numbers are different to all of these.

Nested Query: Single Columns Example 3

empno	firstname	familyname	job	salary	deptno
1234	Sean	Russell	Teacher	50000	10
4567	Jamie	Heaslip	Manager	47000	10
6542	Leo	Cullen	Teacher	45000	10
6675	Abraham	Macken	Designer	34000	45
1238	Brendan	Macken	Technician	25000	20
1555	Sean	O'Brien	Designer	50000	20
1899	Brian	O'Driscoll	Manager	45000	20
2525	Peter	Stringer	Designer	25000	30
1585	Denis	Hickey	Architect	20000	30
1345	Ronan	O'Gara	Manager	29000	30
8888	Paul	O'Connell	Driver	42000	50
7878	Tommy	Bowe	Manager	55000	60

```
SELECT deptname FROM
department
   WHERE deptno !=
ALL(SELECT DISTINCT deptno
     FROM employee WHERE
firstname='Sean');
```

deptno	deptname	office	division	managernumber
10	Training	Lansdown	D1	4567
20	Design	Belfield	D2	1899
30	Implementation	Donnybrook	D1	1345
40	Finance	Ballsbridge	D3	1595

Nested Query: Single Columns Example 4

- Find the name of all employees who earn more money than everybody in the Implementation department.

-
  ```
  SELECT firstname, familyname FROM employee
    WHERE salary > ALL(SELECT salary FROM
    employee JOIN department USING(deptno)
    WHERE deptname='Implementation');
  ```

VIEWS

Views

- SQL provides the ability to use **views** in your schema.
- A view is a **virtual table** based on a query.
- It looks just like a normal table, but it is not stored directly in the database.
- We can allow users to see a subset of one or more tables, without giving them access to the table itself.

Views

- Every view has a **name** and a **SELECT query** that defines it.

- Example

-
  ```
  CREATE VIEW manager AS
    SELECT * FROM employee
    WHERE empno=ANY(SELECT managerno FROM department);
  ```

- After this, ‘managers’ looks just like an ordinary table.
- Changes in the ‘employee’ table will automatically be seen in the ‘manager’ table.