SQL Week 7

Watch video: https://youtu.be/nTWEyZEqPY4?t=1h20m50s

Topics List

- Subqueries
- Views

SQL Week 7
Subqueries

- What is a Subquery?
    - A **subquery** is where we have a complete SELECT statement embedded within another SELECT statement. The results of this *inner* SELECT statement (or *subselect*) are used in the *outer* statement to help determine the contents of the final result.

SQL Week 7
Subqueries

Example

```
SELECT count(copyId) as 'Number of copies'
FROM bookcopy
WHERE isbn =
  (SELECT isbn
    FROM book
    WHERE title ='JavaScript');
```

The inner select returns 1 value for the ISBN (123675432). The condition in the outer select is now WHERE isbn = '123675432'

SQL Week 7
Subqueries

- The statement now becomes:

```
SELECT count(copyId) as 'Number of copies'
FROM bookcopy
WHERE isbn =   '123675432';
```

- The relational operators (=, >, <, etc..) work when the inner select returns 1 value.

SQL Week 7
Subqueries

Example:

```
SELECT title, price
FROM book
WHERE price =
   (SELECT max(price)
    FROM book);
```

- What does this query return?

SQL Week 7
Subqueries

Example:

```
SELECT title, price
FROM book
WHERE price =
   (SELECT max(price)
    FROM book);
```

- This returns the title and price of the most expensive book.

SQL Week 7

Subqueries

- If the inner select returns more than one value, then we use IN with our WHERE clause.
- In the following example, the inner select returns the studentId of all students in the loan table. The outer select will return the names of the student whose studentId value matches one of the returned values.

```
SELECT concat(fname, ' ', lname) as Name
FROM student
WHERE studentid IN
  (SELECT studentid
   FROM loan);
```

SQL Week 7
Subqueries

- The keywords ANY and ALL may be used with subqueries that produce a single column of numbers.
- The following example returns the title and price of the book whose price >= all the prices returned from the book table.

```
SELECT title, price
FROM book
WHERE price >= ALL
  (SELECT price
   FROM book);
```

SQL Week 7
Subqueries

- The following example returns the book title, category and price of all Computing books.

```
SELECT title, category, price
FROM book
WHERE price > ALL
  (SELECT price
   FROM book
   WHERE category = 'Computing');
```

SQL Week 7
Subqueries

- The following example returns the book title, category and price of any book whose price is greater than the price of at least one Computing book.

```
SELECT title, category, price
FROM book
WHERE price > ANY
  (SELECT price
   FROM book
   WHERE category = 'Computing');
```

Topics List

- Subqueries
- Views

SQL Week 7
Views

- What is a View?
    - A **view** is the dynamic result of querying a base relation (table) to produce another relation (result-set). A view is a *virtual relation* that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request.

SQL Week 7

Views

Example

```
CREATE VIEW overduedetails AS
   SELECT concat(fname,' ',lname)  AS Name, title,
            dateOut
FROM bookcopy JOIN book
ON  bookcopy.isbn = book.isbn
JOIN loan
ON bookcopy.copyId = loan.copyId
JOIN student
ON student.studentId = loan.studentId
WHERE dateBack IS NULL;
```

SQL Week 7

Views

- This example creates a view (overduedetails) that returns the details of overdue loans.
- This just creates the definition of the view, to execute the view, we select from the view:

```
SELECT * FROM overduedetails;
```

SQL Week 7
Views

- In MySQL, views are not only query-able but also updatable. It means that you can use the INSERT or UPDATE statement to insert or update rows of the base table through the updatable view. In addition, you can use DELETE statement to remove rows of the underlying table through the view.

SQL Week 7

Views

Example

```
CREATE OR REPLACE view pricedetails AS
 SELECT isbn, title, category, price
 FROM book
  WHERE category = 'computing';
```

- This View (pricedetails) selects the book details for all computing books.

SQL Week 7

Views

SELECT * FROM pricedetails; // Select all records for the view

UPDATE pricedetails
 SET price = price * 1.10; //Increase the price of books by 10%;

SELECT * FROM pricedetails; ; // Select all records for the view. Note the price increase.

SELECT * FROM book; ; // Select all records for the table. Note the price increase.

SQL Week 7
Views

- The WITH CHECK OPTION clause is an optional part of the CREATE VIEW statement. The WITH CHECK OPTION clause prevents you from updating or inserting rows that are not visible through the view. In other words, whenever you update or insert a row of the base table through a view, MySQL ensures that the insert or update operation is conformed with the definition of the view.

SQL Week 7

Views

```
CREATE OR REPLACE view pricedetails AS
 SELECT isbn, title, category, price
 FROM book
 WHERE category = 'computing‘
 WITH CHECK OPTION;
```

```
INSERT INTO pricedetails
VALUES('123456777',‘Biology: A Global Approach',
'Life Science', 90);
```

- This INSERT will cause an error as the category value is not Computing.