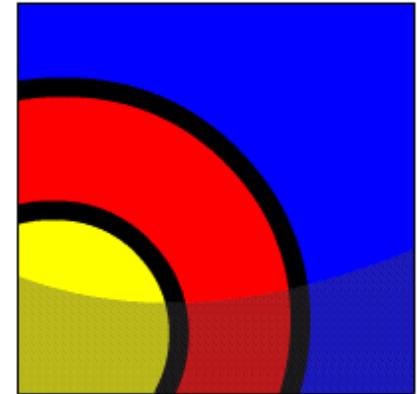


Review of SQL Group Functions and Subqueries

What Will I Learn?

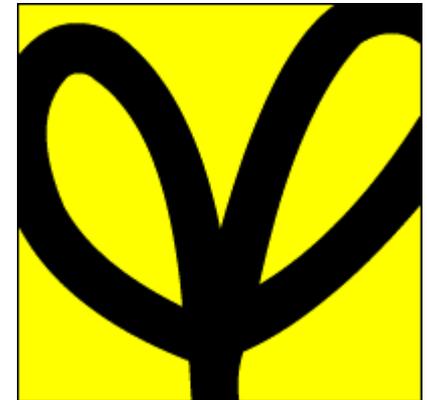
In this lesson, you will review how to:

- Construct and execute an SQL query that uses group functions to determine a sum total, an average amount, and a maximum value
- Construct and execute an SQL query that groups data based on specified criteria
- Construct and execute a query that contains a WHERE clause using a single-row subquery
- Construct and execute a query that contains a WHERE clause using a multiple-row subquery



Why Learn It?

Going over previously learned material reinforces basic concepts and prepares you for more complicated constructs.

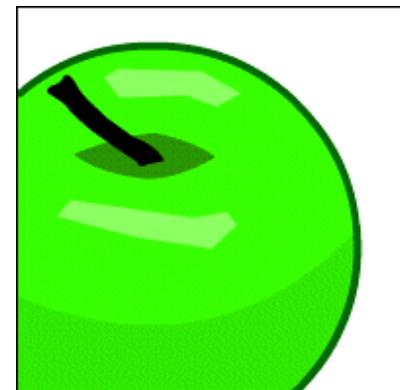


Tell Me/Show Me

Group Functions

These functions operate on a whole table or on a specific grouping of rows to return one result.

`avg`: Used with columns that store numeric data to compute the average, ignoring null values



```
SELECT TO_CHAR(AVG(population), '9,999,999,999.99') average  
FROM wf_countries;
```

AVERAGE
27,130,848.07



Tell Me/Show Me

Group Functions

COUNT: Returns the number of non-null column values or whole rows

```
SELECT COUNT(country_id) "Number of Countries"  
FROM wf_countries;
```

```
SELECT COUNT(*) "Number of Countries"  
FROM wf_countries;
```

Number of Countries
244



Tell Me/Show Me

Group Functions

MIN: Used with columns that store any data type to return the minimum value, ignoring null values

MAX: Used with columns that store any data type to return the maximum value, ignoring null values

SUM: Used with columns that store numeric data to find the total or sum of values, ignoring null values

```
SELECT MIN(lowest_elevation) "All time low"  
FROM wf_countries;
```

All time low
-2555



Tell Me/Show Me

Group Functions

```
SELECT MAX(highest_elevation) "All time high"  
FROM wf_countries;
```

All time high
8850



Tell Me/Show Me

Group Functions

```
SELECT TO_CHAR(SUM(area), '999,999,999,999.99') "Total area"  
FROM wf_countries;
```

Total area
148,148,433.00



Tell Me/Show Me

GROUP BY

Use the `GROUP BY` clause to divide the rows in a table into smaller groups. You can then use the group functions to return summary information for each group.

The `WHERE` clause first excludes rows. The remaining data is divided into groups, and group functions are applied.

```
SELECT region_id, COUNT(country_id)
FROM wf_countries
WHERE region_id < 15
GROUP BY region_id;
```

REGION_ID	COUNT(COUNTRY_ID)
5	15
9	28
11	21
13	8
14	7



Tell Me/Show Me

HAVING

Use the `HAVING` clause to restrict groups. In a query that uses a `GROUP BY` and `HAVING` clause, the rows are first grouped, group functions are applied, and then only those groups matching the `HAVING` clause are displayed.

The following is the syntax:

```
SELECT column, group_function
FROM table
WHERE <restrict rows>
GROUP BY <form subgroups>
HAVING <restrict groups>
ORDER BY <sort rows remaining>
```



Tell Me/Show Me

HAVING

```
SELECT region_id, COUNT(country_id)
FROM wf_countries
WHERE region_id < 15
GROUP BY region_id
HAVING COUNT(country_id) < 20
ORDER BY region_id DESC;
```

REGION_ID	COUNT(COUNTRY_ID)
14	7
13	8
5	15

Tell Me/Show Me

Subqueries

A subquery is a `SELECT` statement that is embedded in a clause of another `SQL` statement. The following is the syntax:

```
SELECT select_list
FROM table
WHERE expression operator
(SELECT select_list FROM table);
```

Single-row subqueries use single-row operators (`>`, `=`, `>=`, `<` `<>`, and `<=`) and return only one row from the subquery.

Tell Me/Show Me

Subqueries (continued)

You can place subqueries in a number of SQL clauses, including the `WHERE` clause, the `HAVING` clause, and the `FROM` clause.

The example shown returns the countries that are bigger than Brazil. The subquery or inner `SELECT` returns one row.

The `ORDER BY` clause is always last.

```
SELECT country_name,  
TO_CHAR(area, '999,999,999,999') area  
FROM wf_countries  
WHERE area >  
      (SELECT area  
       FROM wf_countries  
       WHERE country_name =  
             'Federative Republic of Brazil')  
ORDER BY country_name;
```

COUNTRY_NAME	AREA
Antarctica	14,000,000
Canada	9,984,670
People Republic of China	9,984,670
Russian Federation	17,075,200
United States of America	9,631,420

Tell Me/Show Me

Group Functions and Subqueries

You can use group functions in subqueries because they return a single row. Look at the following example:

```
SELECT country_name
FROM wf_countries
WHERE lowest_elevation =
      (SELECT MIN (lowest_elevation)
       FROM wf_countries);
```

It returns the country with the smallest value for `lowest_elevation` in the `wf_countries` table.

COUNTRY_NAME
Antarctica



Tell Me/Show Me

Group Functions

This example shows that you can use subqueries with joins and other WHERE conditions.

```
SELECT country_name, population
FROM wf_countries
WHERE population =
  (SELECT MAX(population)
   FROM wf_countries c, wf_world_regions wr
   WHERE c.region_id = wr.region_id
   AND wr.region_name = 'Oceania');
```

COUNTRY_NAME	POPULATION
Commonwealth of Australia	20264082



Tell Me/Show Me

Multiple-Row Subqueries

Multiple-row subqueries use multiple-row operators (`IN`, `ANY`, and `ALL`), and return more than one row from the inner query.

You use the `IN` operator when the outer query `WHERE` clause is designed to restrict rows based on a list of values returned from the inner query.



Tell Me/Show Me

Multiple-Row Subqueries (continued)

The following example displays the name, population, and number of airports of each country that has more than one airport.

```
SELECT country_name, population, airports
FROM wf_countries
WHERE country_id IN
      (SELECT country_id
       FROM wf_countries WHERE airports >1);
```

COUNTRY_NAME	POPULATION	AIRPORTS
United Arab Emirates	2602713	35
Republic of Azerbaijan	7961619	45
Republic of Armenia	2976372	16
Commonwealth of Australia	20264082	450
Republic of Austria	8192880	55
Antarctica	0	28
Republic of Botswana	1639833	85



Tell Me/Show Me

ANY and ALL Operators

You use the `ANY` operator when the outer query `WHERE` clause is designed to restrict rows based on any value returned from the inner query.

You use the `ALL` operator when the outer query `WHERE` clause is designed to restrict rows based on all values returned from the inner query.



Tell Me/Show Me

ANY Operator

The following example returns the name, population, and area of any country having an area less than 1,000.

```
SELECT country_name, population, area
FROM wf_countries
WHERE country_id = ANY
      (SELECT country_id
       FROM wf_countries
       WHERE area <1000);
```

COUNTRY_NAME	POPULATION	AREA
Territory of Cocos (Keeling) Islands	574	14
Jay Mayen	-	373
Principality of Liechtenstein	33987	160
Principality of Monaco	32543	2
Republic of Maldives	359008	300



Tell Me/Show Me

ALL Operator

The following example returns the name of each country in all regions that do not start with the letter “A.”

```
SELECT country_name
FROM wf_countries c, wf_world_regions wr
WHERE c.region_id = wr.region_id
AND region_name > ALL
  (SELECT region_name
   FROM wf_world_regions
   WHERE UPPER(region_name) LIKE 'A%');
```

COUNTRY_NAME
United Arab Emirates
Republic of Azerbaijan
Republic of Armenia

Tell Me / Show Me

Terminology

Key terms used in this lesson include:

Group functions

GROUP BY

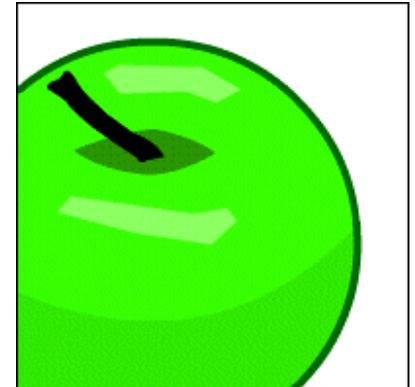
HAVING

Subquery

Multiple row subqueries

ANY

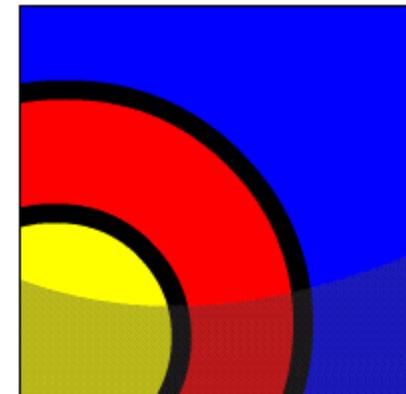
ALL



Summary

In this lesson, you have learned how to:

- Construct and execute an SQL query that uses group functions to determine a sum total, an average amount, and a maximum value
- Construct and execute an SQL query that groups data based on specified criteria
- Construct and execute a query that contains a WHERE clause using a single-row subquery
- Construct and execute a query that contains a WHERE clause using a multiple-row subquery



Try It/Solve It

The exercises in this lesson cover group functions in a SQL query:

- AVG, COUNT, MAX, MIN, and SUM group functions
- GROUP BY clause
- HAVING clause
- Single-row subquery in the WHERE clause
- Multiple-row subquery in the WHERE clause

