Created by Ahsan Arif

COUNT Function

The COUNT function returns the number of rows in a query.

The syntax for the COUNT function is:

SELECT COUNT(*expression*)
FROM tables
WHERE predicates;

Note:

The COUNT function will only count those records in which the field in the brackets is NOT NULL.

For example, if you have the following table called **suppliers**:

Supplier_ID	Supplier_Name	State
1	IBM	CA
2	Microsoft	
3	NVIDIA	

The result for this query will return 3.

Select COUNT(Supplier_ID) from suppliers;

While the result for the next query will only return 1, since there is only one row in the suppliers table where the State field is NOT NULL.

Select COUNT(State) from suppliers;

Simple Example

For example, you might wish to know how many employees have a salary that is above $25,000 / year.

```
SELECT COUNT(*) as "Number of employees"
FROM employees
WHERE salary > 25000;
```

In this example, we've aliased the count(*) field as "Number of employees". As a result, "Number of employees" will display as the field name when the result set is returned.

Example using DISTINCT

You can use the DISTINCT clause within the COUNT function.

For example, the SQL statement below returns the number of unique departments where at least one employee makes over $25,000 / year.

```
SELECT COUNT(DISTINCT department) as "Unique departments"
FROM employees
WHERE salary > 25000;
```

Again, the count(DISTINCT department) field is aliased as "Unique departments". This is the field name that will display in the result set.

Example using GROUP BY

In some cases, you will be required to use a GROUP BY clause with the COUNT function.

For example, you could use the COUNT function to return the name of the department and the number of employees (in the associated department) that make over $25,000 / year.

```
SELECT department, COUNT(*) as "Number of employees"
FROM employees
WHERE salary > 25000
GROUP BY department;
```

Because you have listed one column in your SELECT statement that is not encapsulated in the COUNT function, you must use a GROUP BY clause. The department field must, therefore, be listed in the GROUP BY section.

Performance Tuning

Since the COUNT function will return the same results regardless of what NOT NULL field(s) you include as the COUNT function parameters (ie: within the brackets), you can change the syntax of the COUNT function to COUNT(1) to get better performance as the database engine will not have to fetch back the data fields.

For example, based on the example above, the following syntax would result in better performance:

```
SELECT department, COUNT(1) as "Number of employees"
FROM employees
WHERE salary > 25000
GROUP BY department;
```

Now, the COUNT function does not need to retrieve all fields from the employees table as it had to when you used the COUNT(*) syntax. It will merely retrieve the numeric value of 1 for each record that meets your criteria.

Practice Exercise #1:

Based on the *employees* table populated with the following data, count the number of employees whose salary is over $55,000 per year.

```
CREATE TABLE employees
( employee_number    number(10)        not null,
  employee_name      varchar2(50)      not null,
  salary             number(6),
  CONSTRAINT employees_pk PRIMARY KEY
  (employee_number)
);


INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1001, 'John Smith', 62000);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1002, 'Jane Anderson', 57500);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1003, 'Brad Everest', 71000);

INSERT INTO employees (employee_number, employee_name, salary)
VALUES (1004, 'Jack Horvath', 42000);
```

Solution:

Although inefficient in terms of performance, the following SQL statement would return the number of employees whose salary is over $55,000 per year.

```
SELECT COUNT(*) as "Number of employees"
FROM employees
WHERE salary > 55000;
```

It would return the following result set:

Number of employees
3

A more efficient implementation of the same solution would be the following SQL statement:

```
SELECT COUNT(1) as "Number of employees"
FROM employees
WHERE salary > 55000;
```

Now, the COUNT function does not need to retrieve all of the fields from the table (ie: employee_number, employee_name, and salary), but rather whenever the condition is met, it will retrieve the numeric value of 1. Thus, increasing the performance of the SQL statement.

Practice Exercise #2:

Based on the *suppliers* table populated with the following data, count the number of distinct *cities* in the *suppliers* table:

```
CREATE TABLE suppliers
( supplier_id          number(10)        not null,
  supplier_name        varchar2(50)      not null,
  city                 varchar2(50),
  CONSTRAINT suppliers_pk PRIMARY KEY
  (supplier_id)
);


INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES (5001, 'Microsoft', 'New York');

INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES (5002, 'IBM', 'Chicago');
```

```
INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES (5003, 'Red Hat', 'Detroit');

INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES (5004, 'NVIDIA', 'New York');

INSERT INTO suppliers (supplier_id, supplier_name, city)
VALUES (5005, 'NVIDIA', 'LA');
```

Solution:

The following SQL statement would return the number of distinct *cities* in the *suppliers* table:

```
SELECT COUNT(DISTINCT city) as "Distinct Cities"
FROM suppliers;
```

It would return the following result set:

Distinct Cities
4

Practice Exercise #3:

Based on the *customers* table populated with the following data, count the number of distinct *cities* for each *customer_name* in the *customers* table:

```
CREATE TABLE customers
( customer_id       number(10)      not null,
  customer_name     varchar2(50)    not null,
  city              varchar2(50),
  CONSTRAINT customers_pk PRIMARY KEY
  (customer_id)
);


INSERT INTO customers (customer_id, customer_name, city)
VALUES (7001, 'Microsoft', 'New York');

INSERT INTO customers (customer_id, customer_name, city)
VALUES (7002, 'IBM', 'Chicago');
```

```
INSERT INTO customers (customer_id, customer_name, city)
VALUES (7003, 'Red Hat', 'Detroit');

INSERT INTO customers (customer_id, customer_name, city)
VALUES (7004, 'Red Hat', 'New York');

INSERT INTO customers (customer_id, customer_name, city)
VALUES (7005, 'Red Hat', 'San Francisco');

INSERT INTO customers (customer_id, customer_name, city)
VALUES (7006, 'NVIDIA', 'New York');

INSERT INTO customers (customer_id, customer_name, city)
VALUES (7007, 'NVIDIA', 'LA');

INSERT INTO customers (customer_id, customer_name, city)
VALUES (7008, 'NVIDIA', 'LA');
```

Solution:

The following SQL statement would return the number of distinct *cities* for each *customer_name* in the *customers* table:

```
SELECT customer_name, COUNT(DISTINCT city) as "Distinct Cities"
FROM customers
GROUP BY customer_name;
```

It would return the following result set:

CUSTOMER_NAME	Distinct Cities
IBM	1
Microsoft	1
NVIDIA	2
Red Hat	3