CSE 3104 : DATABASE SESSIONAL

SQL Aggregate Functions

SQL aggregate functions return a single value, calculated from values in a column.

Useful aggregate functions:

- AVG() - Returns the average value
- COUNT() - Returns the number of rows
- FIRST() - Returns the first value
- LAST() - Returns the last value
- MAX() - Returns the largest value
- MIN() - Returns the smallest value
- SUM() - Returns the sum

SQL AVG() Example

We have the following "Orders" table:

O_Id	OrderDate	OrderPrice	Customer
1	2008/11/12	1000	Hansen
2	2008/10/23	1600	Nilsen
3	2008/09/02	700	Hansen
4	2008/09/03	300	Hansen
5	2008/08/30	2000	Jensen
6	2008/10/04	100	Nilsen

Now we want to find the average value of the "OrderPrice" fields.

The AVG() Function

The AVG() function returns the average value of a numeric column.

SQL AVG() Syntax

```
SELECT AVG(column_name) FROM table_name
```

We use the following SQL statement:

```
SELECT AVG(OrderPrice) AS OrderAverage FROM Orders
```

The result-set will look like this:

OrderAverage
950

Now we want to find the customers that have an OrderPrice value higher than the average OrderPrice value.

We use the following SQL statement:

```
SELECT Customer FROM Orders
WHERE OrderPrice>(SELECT AVG(OrderPrice) FROM Orders)
```

The result-set will look like this:

Customer
Hansen
Nilsen
Jensen

The COUNT() Function

The COUNT() function returns the number of rows that matches a specified criteria.

SQL COUNT(column_name) Syntax

The COUNT(column_name) function returns the number of values (NULL values will not be counted) of the specified column:

```
SELECT COUNT(column_name) FROM table_name
```

SQL COUNT(*) Syntax

The COUNT(*) function returns the number of records in a table:

```
SELECT COUNT(*) FROM table_name
```

Now we want to count the number of orders from "Customer Nilsen".

We use the following SQL statement:

```
SELECT COUNT(Customer) AS CustomerNilsen FROM Orders
WHERE Customer='Nilsen'
```

The result of the SQL statement above will be 2, because the customer Nilsen has made 2 orders in total:

CustomerNilsen
2

SQL COUNT(*) Example

If we omit the WHERE clause, like this:

```
SELECT COUNT(*) AS NumberOfOrders FROM Orders
```

The result-set will look like this:

NumberOfOrders
6

which is the total number of rows in the table.

The FIRST() Function

The FIRST() function returns the first value of the selected column.

SQL FIRST() Syntax

```
SELECT FIRST(column_name) FROM table_name
```

Now we want to find the first value of the "OrderPrice" column.

We use the following SQL statement:

```
SELECT FIRST(OrderPrice) AS FirstOrderPrice FROM Orders
```

Tip: Workaround if FIRST() function is not supported:

```
SELECT OrderPrice FROM Orders ORDER BY O_Id LIMIT 1
```

The result-set will look like this:

FirstOrderPrice
1000

The LAST() Function

The LAST() function returns the last value of the selected column.

SQL LAST() Syntax

```
SELECT LAST(column_name) FROM table_name
```

Now we want to find the last value of the "OrderPrice" column.

We use the following SQL statement:

```
SELECT LAST(OrderPrice) AS LastOrderPrice FROM Orders
```

Tip: Workaround if LAST() function is not supported:

```
SELECT OrderPrice FROM Orders ORDER BY O_Id DESC LIMIT 1
```

The result-set will look like this:

LastOrderPrice
100

The MAX() Function

The MAX() function returns the largest value of the selected column.

SQL MAX() Syntax

```
SELECT MAX(column_name) FROM table_name
```

Now we want to find the largest value of the "OrderPrice" column.

We use the following SQL statement:

```
SELECT MAX(OrderPrice) AS LargestOrderPrice FROM Orders
```

The result-set will look like this:

LargestOrderPrice
2000

The MIN() Function

The MIN() function returns the smallest value of the selected column.

SQL MIN() Syntax

```
SELECT MIN(column_name) FROM table_name
```

Now we want to find the smallest value of the "OrderPrice" column.

We use the following SQL statement:

```
SELECT MIN(OrderPrice) AS SmallestOrderPrice FROM Orders
```

The result-set will look like this:

SmallestOrderPrice
100

The SUM() Function

The SUM() function returns the total sum of a numeric column.

SQL SUM() Syntax

```
SELECT SUM(column_name) FROM table_name
```

Now we want to find the sum of all "OrderPrice" fields".

We use the following SQL statement:

```
SELECT SUM(OrderPrice) AS OrderTotal FROM Orders
```

The result-set will look like this:

OrderTotal
5700

The GROUP BY Statement

The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.

SQL GROUP BY Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
```

Now we want to find the total sum (total order) of each customer.

We will have to use the GROUP BY statement to group the customers.

We use the following SQL statement:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
```

The result-set will look like this:

Customer	SUM(OrderPrice)
Hansen	2000
Nilsen	1700
Jensen	2000

GROUP BY More Than One Column

We can also use the GROUP BY statement on more than one column, like this:

```
SELECT Customer,OrderDate,SUM(OrderPrice) FROM Orders
GROUP BY Customer,OrderDate
```

The HAVING Clause

The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

SQL HAVING Syntax

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name
HAVING aggregate_function(column_name) operator value
```

Now we want to find if any of the customers have a total order of less than 2000.

We use the following SQL statement:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
GROUP BY Customer
HAVING SUM(OrderPrice)<2000
```

The result-set will look like this:

Customer	SUM(OrderPrice)
Nilsen	1700

Now we want to find if the customers "Hansen" or "Jensen" have a total order of more than 1500.

We add an ordinary WHERE clause to the SQL statement:

```
SELECT Customer,SUM(OrderPrice) FROM Orders
WHERE Customer='Hansen' OR Customer='Jensen'
GROUP BY Customer
HAVING SUM(OrderPrice)>1500
```

The result-set will look like this:

Customer	SUM(OrderPrice)
Hansen	2000
Jensen	2000