SQL Basics

SQL provides the following GROUP FUNCTIONS

SUM – Provides the sum of the values in a column across many rows

AVG - Provides the average of the values in a column across many rows

COUNT – Provides a count of how many rows have a value in a column, counted across many rows

MIN - Provides the lowest value in a column across many rows

MAX - Provides the highest value in a column across many rows

GROUP FUNCTIONS

- SUM, AVG must only be used with NUMERIC columns
- MIN, MAX work with any type of column
- COUNT can be used with any column, or with a (*) to simply count rows

GROUP FUNCTIONS

- Group functions require SQL to create an interim answer set, and then process the group function against the interim answer set, delivering a final answer set that contains only the final total for the function.
- When you combine a GROUP FUNCTION with a WHERE clause, keep in mind that the WHERE clause simply reduces the number of rows in the INTERIM answer set before the GROUP function does its calculation.

Examples –

```
select COUNT(*) as 'Total'
      from nwEmployees
select COUNT(Distinct Country) as 'Countries'
      from nwCustomers
select SUM(UnitPrice) as 'Total Price'
      from nwProducts
select MAX(UnitPrice) as 'High Price'
      from nwProducts
select MIN(UnitPrice) as 'Low Price'
      from nwProducts
      where UnitPrice > 0
select AVG(UnitsInStock) as 'Average Inventory'
      from nwProducts
```

Count Examples –

```
select Country, COUNT(*) as 'Total'
       from nwCustomers
       GROUP BY Country

select Country, COUNT(Country) as 'Total'
       from nwCustomers
       GROUP BY Country
```

Why are these the same?

```
SELECT OrderID, ProductID, UnitPrice, Quantity
    FROM nwOrderDetails
    WHERE OrderID in (10248, 10249, 10250,10251);
```

OrderID	ProductID	UnitPrice	Quantity
10248	11	14.00	12
10248	42	9.80	10
10248	72	34.80	5
10249	14	18.60	9
10249	51	42.40	40
10250	41	7.70	10
10250	51	42.40	35
10250	65	16.80	15
10251	22	16.80	6
10251	57	15.60	15
10251	65	16.80	20

```
SELECT OrderID, ProductID, SUM(UnitPrice), SUM(Quantity)
  FROM nwOrderDetails
  WHERE OrderID in (10248, 10249, 10250,10251);
```

OrderID	ProductID	sum(UnitPrice)	Sum(Quantity)
10248	11	235.70	177

```
SELECT OrderID, ProductID, SUM(UnitPrice), SUM(Quantity)
  FROM nwOrderDetails
  WHERE OrderID in (10248, 10249, 10250,10251)
  GROUP BY OrderID;
```

OrderID	ProductID	SUM(UnitPrice)	SUM(Quantity)
10248	11	58.60	27
10249	14	61.00	49
10250	41	66.90	60
10251	22	49.20	41

GROUP BY

- Group functions process against an interim answer set to return a value across many rows.
- Using a GROUP BY clause enables SQL to provide subtotals. The GROUP BY tells SQL to perform the group function against a subset of rows in the interim answer set and provide a total for each subset of rows.

IMPORTANT RULE

When using a GROUP BY every column in the SELECT statement must either be a GROUP FUNCTION or a COLUMN that you are grouping by.

Strict Mode = Fail with error code

Not Strict Mode = meaningless data in answer set

From which supplier does Northwinds carry the most inventory?

```
SELECT SupplierID, SUM(UnitsInStock)AS 'Inventory'
   FROM nwProducts
   GROUP BY SupplierID
   ORDER BY SUM(UnitsInStock) DESC;
```

In which month in 2014 did Northwinds ship the most orders?

```
SELECT  EXTRACT(YEAR FROM ShippedDate) AS 'ShippedYear',
   EXTRACT(MONTH FROM shippeddate) AS 'ShippedMonth',
   COUNT(OrderID) AS 'Orders'
   FROM nwOrders
   WHERE EXTRACT(YEAR FROM ShippedDate) = '2014'
   GROUP BY EXTRACT(MONTH FROM ShippedDate)
   ORDER BY COUNT(orderid) DESC;
```

HAVING

- Is simply like a WHERE clause against the answer set when you use a GROUP BY

Examples –

```
select Country, COUNT(*) as 'Total'
       from nwCustomers
       GROUP BY Country
```

In which countries does Northwinds have more than five customers?

```
select Country, COUNT(*) as 'Total'
       from nwCustomers
       GROUP BY Country
       HAVING  COUNT(*) > 5
       order by 2 desc
```

SubQuery – two styles of use

1. Subquery in WHERE. A query within a query. The answer set to an “inner” query is used as a predicate in a where clause in the “outer” query.
2. Subquery in SELECT or FROM. A derived value or table. The answer set to an “inner” query is used as a table in a select or from clause.

Subquery must be enclosed in parentheses.

SubQuery – two styles of use

Subquery in WHERE. A query within a query. The answer set to an “inner” query is used as a predicate in a where clause in the “outer” query.

- The subquery must return only one column.
- If the outer query WHERE clause contains an “equals” condition, the subquery must return ONE row.
- If the outer query WHERE clause contains an “in” condition, the subquery may return multiple rows, presented as a list of values.
- The Subquery is embedded within parentheses
- Outer and inner queries can hit two different tables

Examples – Find the productID, name and unit price of the highest priced product Northwinds sells.

```
select ProductID, ProductName, UnitPrice
    from nwProducts
    where UnitPrice = (

        select MAX(UnitPrice)
              from nwProducts )
```

Note that with the “equals” condition, the inner query returns only one value (one row, one column)

Examples – Find the CustomerID and the OrderID for all orders with more than 100 units sold.

```
select CustomerID, OrderID
    from nwOrders
    where OrderID in (

        select OrderID
              from nwOrderDetails
              where Quantity > 100 )

    Order by CustomerID
```

- Note that with the “in” condition, the inner query returns many values (many rows, one column) as a list
- Uses Two different tables

SubQuery – two styles of use

A derived value/table. The answer set to an “inner” query is used as a table in a select or from clause.

When used in the SELECT, the subquery must return a single value.

Used in the FROM, the subquery serves as a derived table, like a view or temp table. Such derived tables must have an alias.

Examples – Create a list of all orders with fewer than 100 items sold.

```
SELECT OrderID
   FROM (SELECT OrderID, COUNT(quantity)
            FROM  nwOrderDetails
            GROUP BY OrderID
            HAVING COUNT(quantity) < 100) AS DetailCount;
```

Co-Related SubQuery

Inner versus Outer query.

The inner query is re-executed with each row returned by the outer query.

Can yield very slow performance issues.

Example – Select all the employees where the employee's home city is equal to their orders' ship city.

```
SELECT OrderID, ShipCity, CustomerID, O.EmployeeID
    FROM nwOrders O
    WHERE EmployeeID IN
    (SELECT EmployeeID FROM nwEmployees E
        WHERE E.City = O.ShipCity);
```