PROG8080
GROUP BY and HAVING

Glenn Paulley

Fall 2015

GROUP BY

- GROUP BY, in combination with aggregate functions, can be used to group rows and summarize the results of aggregate functions for each group
- Once GROUP BY is used, the expressions in a query's SELECT list are restricted to:
  - Expressions specified in the GROUP BY clause
  - Aggregate functions (or other computations based on their results)

What does GROUP BY do?

- GROUP BY partitions the input rows (FROM and WHERE clauses) into groups that have the identical set of values for the GROUP BY expressions specified in the query
- Once all of the input rows are partitioned, the server then computes the result of the aggregate functions for each partition
- Each partition then generates a single row in the output

Using GROUP BY

```
SELECT studentNumber, SUM(amount)
FROM dbo.Payment
GROUP BY studentNumber;
```

or

```
SELECT studentNumber,
    '$' + CONVERT( CHAR(12),
            CAST( SUM(amount) AS money ), 1 )
             AS "Invoice Total"
FROM dbo.Payment
GROUP BY studentNumber;
```

GROUP BY result

studentNumber	Invoice Total
1114453	$ 1,000.00
1335314	$ 10,000.00
2286425	$ 1,000.00
2642726	$ 13,159.00
2826147	$ 1,000.00
3244449	$ 5,291.44
3868247	$ 13,159.00
6644710	$ 1,159.00
7252620	$ 5,291.44
7677479	$ 2,000.00
7681752	$ 1,000.00
7826662	$ 6,791.44
8431710	$ 5,291.44
8588766	$ 1,000.00
8866782	$ 3,000.00

What forms a GROUP?

- Groups are formed from unique values of the combination of grouping columns in the GROUP BY clause
  - For the purposes of comparing column or expression values, NULL is semantically equivalent to NULL, as in SELECT DISTINCT
- GROUP BY X
  - Groups are formed based on unique values of X
- GROUP BY X, Y
  - Groups are formed based on unique combinations of values of X and Y for the same row

Empty input and GROUP BY

- If a query's intermediate result is empty,
  - and that query contains GROUP BY
  - then the GROUP BY query also returns the empty set

- **This is not the case for queries that involve aggregation but don't have a GROUP BY clause**

HAVING

- HAVING can be used to limit the results that appear in groups created with GROUP BY
- HAVING is similar to a WHERE clause
  - WHERE restricts the rows created by the FROM clause
  - HAVING restricts the groups created from the GROUP BY clause
- HAVING is often used with aggregate functions
  - You cannot use an aggregate function in a WHERE clause because at the point of applying the WHERE conditions, the groups have yet to be formed

GROUP BY and HAVING

```
SELECT studentNumber,
    '$' + CONVERT( CHAR(12),
            CAST( SUM(amount) AS money ), 1 )
            AS "Invoice Total"
FROM dbo.Payment
GROUP BY studentNumber
HAVING COUNT(*) > 1;
```

- Determine the SUM of payments for specific students but only for students who have paid over more than one payment

GROUP BY and HAVING result

studentNumber	Invoice Total
2642726	$ 13,159.00
7677479	$ 2,000.00
7826662	$ 6,791.44
8431710	$ 5,291.44
8866782	$ 3,000.00

ORDER OF CLAUSES

- If a query contains a WHERE clause, you must put the WHERE clause before GROUP BY and HAVING
- ORDER BY follows GROUP BY and the optional HAVING clause

```
SELECT studentNumber,
    '$' + CONVERT( CHAR(12),
            CAST( SUM(amount) AS money ), 1 )
            AS "Invoice Total"
FROM dbo.Payment
WHERE studentNumber <> 8431710
GROUP BY studentNumber
HAVING COUNT(*) > 1
ORDER BY 2 DESC;
```

GROUP BY, HAVING, WHERE and ORDER BY

studentNumber	Invoice Total
2642726	$ 13,159.00
7826662	$ 6,791.44
8866782	$ 3,000.00
7677479	$ 2,000.00

Expressions with GROUP BY

- One cannot specify an expression in a query's SELECT list that is NOT an aggregate function unless the identical expression appears in the query's GROUP BY clause
- For example:
  - SELECT X, SUM(Y) FROM T GROUP BY X – permitted
  - SELECT X, SUM(Y) FROM T – illegal
  - SELECT X, Y, SUM(Z) FROM T GROUP BY X – illegal
  - SELECT X, SUM(Z) FROM T GROUP BY X, Y – permitted