SQL Unit 5
Aggregation, GROUP BY, and HAVING

Ryan Nixon

Queries So Far

```
SELECT FUNCTION(column1),
   column2 AS alias
FROM table AS table1
INNER JOIN table AS table2
   ON (table2.pkey = table1.fkey)
OUTER JOIN table AS table3
   ON (table3.fkey = table2.pkey)
WHERE column2 = 'string'
LIMIT rowcount
```

Queries So Far

1. Retrieve data **from** table
2. Add to that data from **join**ed tables
3. Filter results **where** certain conditions match
4. Sort results by the desired **order**ing
5. Return only a **limit**ed number of **select** columns

The Problem With Aggregates

- COUNT, MAX, MIN, AVG, SUM, etc.
- Calculate a result off of an entire column
- Return a single row based off of all values in that column
- But what if you want to see results off of many groups in that column?

Introducing Grouping

- Grouping, using GROUP BY, fixes that
- Grouping allows aggregate columns to be separated according to the values in other columns
- It is best explained with an example

Introducing Grouping

```
SELECT assn_id, MAX(points) AS max
FROM grade
GROUP BY assn_id
ORDER BY assn_id
```

grade			
id	student_id	assn_id	points
1	1	1	5.00
2	6	3	27.00
3	6	1	9.00
4	5	2	13.50
5	3	3	25.00

- Returns the maximum number of points in the table, grouped by assignment

assn_id	max
1	9.00
2	13.50
3	27.00

- Alternately: Returns each assignment with its maximum number of points

Introducing Grouping

- The ORDER BY statement is not required, but a general best practice
- Most databases will sort according to the grouped columns, ascending. You should not assume this

Grouping With Nulls

- Note that GROUP BY does not ignore Nulls, but the functions in the SELECT such as COUNT or SUM still will
- This may cause some functions to return 0 where they may be expected to not appear in the results

Grouping With Nulls

```
SELECT last,
   COUNT(address_id) AS count
FROM person
GROUP BY last
```

person			
id	first	last	address_id
1	Beau	Bridges	NULL
2	Jeffery	Bridges	NULL
3	Orville	Wright	43
4	Wilbur	Wright	13
5	Simon	Tam	27
6	River	Tam	27

- Returns the number of non-null addresses in the table, grouped by last name

last	count
Bridges	0
Wright	2
Tam	2

- Note that "Bridges" still appears even though all of the associated addresses are null

Grouping Multiple Columns

- There is no limit to how many columns you may group by
- `GROUP BY make, model` might be excellent if you are looking for a number of car sales grouped first by the manufacturer, then by the model of the car.
- `GROUP BY make, model, type` would further categorize the results by body type

Non-Grouped Columns

- Non-Aggregates such as fields or row-level functions represent individual rows rather than the collection of all rows
- Because of this, they may invalidate the aggregate data, especially when duplicates occur
- Let's demonstrate this using the previous example

Non-Grouped Columns

```
SELECT first, last,
   COUNT(address_id) AS count
FROM person
```

person			
id	first	last	address_id
1	Beau	Bridges	NULL
2	Jeffery	Bridges	NULL
3	Orville	Wright	43
4	Wilbur	Wright	13
5	Simon	Tam	27
6	River	Tam	27

- Without a Group By, COUNT will return a single row. The database will then try to figure out how to fill the other 2 columns

first	last	count
River	Tam	4

- Does River Tam actually have 4 addresses?

Non-Grouped Columns

```
SELECT first, last,
   COUNT(address_id) AS count
FROM person
GROUP BY last
```

person			
id	first	last	address_id
1	Beau	Bridges	NULL
2	Jeffery	Bridges	NULL
3	Orville	Wright	43
4	Wilbur	Wright	13
5	Simon	Tam	27
6	River	Tam	27

- Grouped by only last name, the counts are still not accurate as the database must still fill in the blanks
- Orville Wright now has 2 addresses. So does River Tam

first	last	count
Jeffery	Bridges	0
Wilbur	Wright	2
River	Tam	2

Non-Grouped Columns

```
SELECT first, last,
   COUNT(address_id) AS count
FROM person
GROUP BY last, first
```

person			
id	first	last	address_id
1	Beau	Bridges	NULL
2	Jeffery	Bridges	NULL
3	Orville	Wright	43
4	Wilbur	Wright	13
5	Simon	Tam	27
6	River	Tam	27

- Grouped by both last and first name, all rows are unique again and the counts match. The database does not have to fill in blanks

first	last	count
Beau	Bridges	0
Jeffery	Bridges	0
Orville	Wright	1
Wilbur	Wright	1
Simon	Tam	1
River	Tam	1

Non-Grouped Columns

- When grouping, always remember that the GROUP BY line must contain all non-aggregates or unexpected results will occur
- Be careful - Only MS SQL errors when columns are grouped incorrectly. Other databases will produce the odd results from the examples

Filtering Groups With HAVING

- When you need to filter aggregate fields you can use the HAVING clause
- HAVING is only able to be used when using GROUP BY
- It works exactly like the WHERE clause

Filtering Groups With HAVING

```
SELECT assn_id, MAX(points) AS max
FROM grade
GROUP BY assn_id
HAVING MAX(points) > 10
```

grade

id	student_id	assn_id	points
1	1	1	5.00
2	6	3	27.00
3	6	1	9.00
4	5	2	13.50
5	3	3	25.00

- Filter the results off of the boolean condition provided in HAVING

assn_id	max
2	13.50
3	27.00

- Note that HAVING can also filter non-aggregate columns, just like WHERE
- Aliases also may not be used in HAVING

But What About WHERE Clauses?

- Rule of thumb: When filtering aggregates, use HAVING. When filtering fields, use WHERE
- WHERE does not filter aggregates, and HAVING may only be used in GROUP BY
- Think of this separation more as an organizational benefit

A few examples from the class database...

Queries Now

```
SELECT FUNCTION(column1),
   column2 AS alias
FROM table AS table1
INNER JOIN table AS table2
   ON (table2.pkey = table1.fkey)
OUTER JOIN table AS table3
   ON (table3.fkey = table2.pkey)
WHERE column2 = 'string'
GROUP BY column1, column2
HAVING column1 > number
LIMIT rowcount
```

Queries Now

1. Retrieve data **from** table
2. Add to that data from **join**ed tables
3. Filter results **where** certain conditions match
4. **Group** results by specified columns
5. Filter results **having** certain conditions
6. Sort results by the desired **order**ing
7. Return only a **limit**ed number of **select** columns

Reminders

- It should be obvious by now that learning SQL is a very hands-on activity. Practice and experimentation are key!
- Assignment 5 up tonight. Due 10/22
- Lab time Wednesday in SSB 172