Aggregate Functions

CSC 256, SQL Programming

Aggregate Functions

- Aggregate functions operation on a set of values and return a single value.
- Aggregate functions ignore NULL values.
- Aggregate functions return NULL when no rows are selected.
- Common aggregate functions:
  - COUNT
  - SUM
  - MIN
  - MAX
  - AVG
- The Chinook database is used for the examples.

COUNT

- The COUNT function returns the number of items in a set.
- Example, count the number of rows in a table:

```
select count(*)
from "Genre";
```

  - Note: * is wildcard to count number of rows

- Example, count with an expression:

```
select count("Company")
from "Customer";
```

Aggregates and DISTINCT

- The DISTINCT keyword can be used in an aggregate function.
- Example:

```
select count(distinct "GenreId")
from "Track";
```

SUM

- The SUM function returns the sum of all the values in a set.
- Example:

```
select sum("UnitPrice")
from "Track";
```

MIN, MAX, and AVG

- The remaining common aggregate functions work similarly:
  - The MIN function returns the minimum value in a set.
  - The MIN function returns the maximum value in a set.
  - The AVG function returns the average value in a set.
- Example:

```
select
  min("UnitPrice"),
  max("UnitPrice"),
  avg("UnitPrice")
from "Track";
```

- There are many more aggregate functions; refer to the documentation.

GROUP BY

- The GROUP BY clause groups rows based on values of one or more columns; it returns one row for each group.
- Basic syntax:

```
SELECT
  column1,
  column2,
  aggregate_function(column3)
FROM
  table_name
GROUP BY
  column1,
  column2
```

- Note: with group by the aggregates work on groups instead of rows; each group gets a unique row in the output

GROUP BY Examples

- Example, average length of track in minutes per Composer:

```
select "Composer", avg("Milliseconds" / 60000.0)
from "Track"
group by "Composer";
```

- Common mistake: not using an aggregate function per group:

```
select "Composer", "UnitPrice"
from film
group by "Composer"
```

  - We did not specify how to convert a set of prices to a single value.

GROUP BY Example

- Example, grouping by two columns:

```
select
  "Composer",
  "AlbumId",
  avg("Milliseconds" / 60000.0)
from "Track"
group by "Composer", "AlbumId"
order by "Composer", "AlbumId";
```

HAVING

- The HAVING clause filters on groups before the aggregate function is applied
- Basic syntax:

```
SELECT
  column1,
  column2,
  aggregate_function(column3)
FROM
  table_name
GROUP BY
  column1,
  column2
HAVING
  condition;
```

HAVING Example

- Example, average track length greater than 8 minutes grouped by composer:

```
select
  "Composer",
  avg("Milliseconds" / 60000.0) as length
from "Track"
group by "Composer"
having avg("Milliseconds" / 60000.0) > 8
order by length desc;
```

CASE Expressions and Aggregates

- CASE expressions allow us to define our own groups based on conditional logic
- Example:

```
select
  case
    when "Milliseconds" < 60000 then 'short'
    when "Milliseconds" between 60000 and 300000 then '
    else 'long'
  end,
  count(*)
from "Track"
group by
  case
    when "Milliseconds" < 60000 then 'short'
    when "Milliseconds" between 60000 and 300000 then '
    else 'long'
```

GROUP BY Shorthand

- We can refer to the columns of the result set by position. That is, 1 corresponds to the first field (regardless of name), 2 the second and so on. This works for GROUP BY and ORDER BY.
- This is generally ill-advised for various reasons.
- Example:

```
select
  case
    when "Milliseconds" < 60000 then 'short'
    when "Milliseconds" between 60000 and 300000 then '
    else 'long'
  end,
  count(*)
from "Track"
group by 1;
```

CASE Expressions and Counting

- We can use COUNT and CASE to count arbitrary things
- Example:

```
select
  count(
    case when "GenreId" in (1, 8) then 1 else null end
  ) as "Rock & Reggae",
  count(
    case when "GenreId" not in (1, 8) then 1 else null
  ) as "Everything Else",
  count(*) as total
from "Track";
```

CASE Expressions and Counting (continued)

- A common alternative is to use SUM and CASE to count arbitrary things
- Example:

```
select
  sum(
    case when "GenreId" in (1, 8) then 1 else 0 end
  ) as "Rock & Reggae",
  sum(
    case when "GenreId" not in (1, 8) then 1 else 0 end
  ) as "Everything Else",
  count(*) as total
from "Track";
```