Aggregate Functions

Aggregate Functions

- Also called set functions
- They operate on group of values to produce a single, summarizing value
- Aggregates are applied to a set of rows that can be:
  - All the rows in a table
  - Only those rows specified by a `WHERE` clause
  - Those rows created by a `GROUP BY` clause

Non-aggregate queries process the rows one by one. Each row is processed independently and put into the result.

Aggregate queries do something completely differently – it takes a tables as a whole and constructs new rows from it.

Aggregate Functions

- We can count the number of records that match a certain criteria –
- `COUNT()` – the number of rows in a table
- `COUNT(`*`value`*`)` – the number of non-null values in *value*

Works with all datatypes

- We can find the lowest (minimum) value or the highest (maximum) value –`MIN(`*`value`*`)` / `MAX(`*`value`*`)`

Works with character, numeric and datetime

- We can add the values in a specific column that match a certain criteria—`SUM(`*`value`*`)`
- We can calculate the average -- `AVG(`*`value`*`)`

Only work with Numeric Types

All aggregate functions except Count(value) ignore NULLS

Aggregate Functions

- Aggregates return new result sets
- The result sets have no defined name
- The DBMS will return a name that is defined in the SELECT clause

```
SELECT COUNT( DISTINCT price )
FROM titles
```

COUNT(DISTINCT price)
10

- Use **ALIAS** to provide more meaningful names to your result set

Aggregate functions

- Aggregate functions ignore NULLS!
- An aggregate function can **NOT** appear in a WHERE clause

```
SELECT title_id
FROM books
WHERE page_count = MAX (page_count);
```

Aggregate

- You can **NOT** mix non-aggregate with aggregate in a SELECT clause

Non-Aggregate

Aggregate

```
SELECT title_id, MAX(page_count)
FROM books;
```

Aggregate functions

- You can **NOT** nest aggregate functions

Aggregate Nested Aggregate

```
SELECT SUM(AVG(sales))
FROM titles;
```

- You can not use subqueries in aggregate expressions

Aggregate

```
SELECT AVG(SELECT price FROM titles);
```

Subquery

Combining Aggregate Functions

- SELECT statements **can** include as many of the aggregate functions as required

```
SELECT
COUNT(*) AS 'Num_Books',
AVG(price) AS 'Avg_Price',
MAX(price) AS 'Highest_Price'
FROM titles;
```