**Aggregates** – calculations performed on multiple rows of a table

**SELECT COUNT(<column>)**

**FROM fake\_apps;**

Counts number of non-empty (NOT NULL) values in <column>. Can use \* to select all columns.

**SELECT COUNT(\*)**

**FROM fake\_apps**

**WHERE price = 0;**

Counts number of non-empty rows in table where (price = 0).

**SELECT SUM(downloads)**

**FROM fake\_apps;**

Adds all values in downloads column.

**SELECT MAX(downloads)**

**FROM fake\_apps;**

Returns the maximum value in downloads column from fake\_apps.

**SELECT MIN(downloads)**

**FROM fake\_apps;**

Returns the minimum value in downloads column.

**SELECT AVG(downloads)**

**FROM fake\_apps;**

Returns the average value from downloads column.

**SELECT ROUND(price, 2)**

**FROM fake\_apps;**

Returns price column with each value rounded to 2 decimal places.

­­**SELECT ROUND(AVG(price), 2)**

**FROM fake\_apps;**

Returns the rounded average of price values to 2 decimal places.

**SELECT prices, COUNT(\*) FROM fake\_apps**

**GROUP BY price;**

Queries prices and COUNT(\*), but groups the COUNT(\*) results into frequencies by each unique price.

**SELECT prices, COUNT(downloads)**

**FROM fake\_apps**

**WHERE downloads > 20000**

**GROUP BY price;**

Queries prices and number of non-empty download values if (download > 20000) and grouped by price.

**SELECT year, genre, COUNT(name)**

**FROM movies**

**GROUP 1, 2**

**HAVING COUNT(name) > 10;**

Queries year, genre, COUNT(name) grouped by (year + genre), but the (year + genre) must have COUNT(name) value of > 10. HAVING is a conditional statement, like WHERE, but for GROUP BY clauses. They go before ORDER BY and LIMIT.

