Reporting with SQL Cheatsheet

Ordering Columns

Ordering by a single column criteria:

```
SELECT * FROM <table name> ORDER BY <column> [ASC|DESC];
```

`ASC` is used to order results in ascending order.

`DESC` is used to order results in descending order.

Examples:

```
SELECT * FROM books ORDER BY title ASC;
SELECT * FROM products WHERE name = "Sonic T-Shirt" ORDER B
SELECT * FROM users ORDER BY signed_up_on DESC;
SELECT * FROM countries ORDER BY population DESC;
```

Ordering by multiple column criteria:

```
SELECT * FROM <table name> ORDER BY <column> [ASC|DESC],
                                    <column 2> [ASC|DESC],
                                    ...,
                                    <column n> [ASC|DESC];
```

Ordering is prioritized left to right.

Examples:

```
SELECT * FROM books ORDER BY    genre ASC,
                                title ASC;

SELECT * FROM books ORDER BY    genre ASC,
                                year_published DESC;

SELECT * FROM users WHERE email LIKE "%@gmail.com"
                    ORDER BY    last_name ASC,
                                first_name ASC;
```

Limiting Results

SQLite, PostgreSQL and MySQL

To limit the number of results returned, use the `LIMIT` keyword.

```
SELECT <columns> FROM <table> LIMIT <# of rows>;
```

MS SQL

To limit the number of results returned, use the `TOP` keyword.

```
SELECT TOP <# of rows> <columns> FROM <table>;
```

Oracle

To limit the number of results returned, use the `ROWNUM` keyword in a `WHERE` clause.

```
SELECT <columns> FROM <table> WHERE ROWNUM <= <# of rows>;
```

Paging Through Results

SQLite, PostgreSQL and MySQL

To page through results you can either use the `OFFSET` keyword in conjunction with the `LIMIT` keyword or just with LIMIT alone.

```
SELECT <columns> FROM <table> LIMIT <# of rows> OFFSET <ski
SELECT <columns> FROM <table> LIMIT <skipped rows>, <# of r
```

MS SQL and Oracle

To page through results you can either use the `OFFSET` keyword in conjunction with the `FETCH` keyword. Cannot be used with `TOP`.

```
SELECT <columns> FROM <table> OFFSET <skipped rows> ROWS FE
```

Syntax definitions

- **Keywords**: Commands issued to a database. The data presented in queries is unaltered.
- **Operators**: Performs comparisons and simple manipulation
- **Functions**: Presents data differently through more complex manipulation
- **Arguments** or **Parameters**: Values passed in to functions.

A function looks like:

```
<function name>(<value or column>)
```

Examples:

```
SELECT UPPER("Andrew Chalkley");
SELECT UPPER(name) FROM passport_holders;
```

Concatenating Strings

SQLite, PostgreSQL and Oracle

Use the concatenation operator `||`.

```
SELECT <value or column> || <value or column> || <value or
```

MS SQL

Use the concatenation operator `+`.

```
SELECT <value or column> + <value or column> + <value or co
```

MySQL, PostgreSQL and MS SQL

Use the `CONCAT()` function.

```
SELECT CONCAT(<value or column>, <value or column>, <value
```

Finding Length of Strings

To obtain the length of a value or column use the `LENGTH()` function.

```
SELECT LENGTH(<value or column>) FROM <tables>;
```

Changing the Case of Strings

Use the `UPPER()` function to uppercase text.

```
SELECT UPPER(<value or column>) FROM <table>;
```

Use the `LOWER()` function to lowercase text.

```
SELECT LOWER(<value or column>) FROM <table>;
```

Create Excerpts with Substring

To create smaller strings from larger piece of text you can use the `SUBSTR()` funciton or the substring function.

```
SELECT SUBSTR(<value or column>, <start>, <length>) FROM <t
```

- **<start>** : Specifies where to start in the string
    - if is 0 (zero), then it is treated as 1.
    - if is positive, then the function counts from the beginning of string to find the first character.
    - if is negative, then the function counts backward from the end of string.
- **<finish>** : length of the desired substring

```
SELECT SUBSTR('abcdefg', 3,4);
```

OUTPUT: cdef

```
SELECT SUBSTR('abcdefg', -5,4);
```

OUTPUT: cdef

Replacing Portions of Text

To replace piece of strings of text in a larger body of text you can use the `REPLACE()` function.

```
SELECT REPLACE(<original value or column>, <target string>,
```

Counting Results

To count rows you can use the `COUNT()` function.

```
SELECT COUNT(*) FROM <table>;
```

To count unique entries use the `DISTINCT` keyword too:

```
SELECT COUNT(DISTINCT <column>) FROM <table>;
```

To count aggregated rows with common values use the `GROUP BY` keywords:

```
SELECT COUNT(<column>) FROM <table> GROUP BY <column with c
```

Obtaining Totals

To total up numeric columns use the `SUM()` function.

```
SELECT SUM(<numeric column) FROM <table>;
```

```
SELECT SUM(<numeric column) AS <alias> FROM <table>
                                       GROUP BY <another co
                                       HAVING <alias> <oper
```

Calculating Averages

To get the average value of a numeric column use the `AVG()` function.

```
SELECT AVG(<numeric column>) FROM <table>;
SELECT AVG(<numeric column>) FROM <table> GROUP BY <other c
```

Finding the Maximum and Minimum Values

To get the maximum value of a numeric column use the `MAX()` function.

```
SELECT MAX(<numeric column>) FROM <table>;
SELECT MAX(<numeric column>) FROM <table> GROUP BY <other c
```

To get the minimum value of a numeric column use the `MIN()` function.

```
SELECT MIN(<numeric column>) FROM <table>;
SELECT MIN(<numeric column>) FROM <table> GROUP BY <other c
```

Mathematical Operators

- `*` Multiply
- `/` Divide
- `+` Add
- `-` Subtract

```
SELECT <numeric column> <mathematical operator> <numeric va
```

Up-to-the-Minute Dates and Times

SQLite

To get the current date use: `DATE("now")`

To get the current time use: `TIME("now")`

To get the current date time: `DATETIME("NOW")`

MS SQL

To get the current date use: `CONVERT(date, GETDATE())`

To get the current time use: `CONVERT(time, GETDATE())`

To get the current date time: `GETDATE()`

MySQL

To get the current date use: `CURDATE()`

To get the current time use: `CURTIME()`

To get the current date time: `NOW()`

Oracle and PostgreSQL

To get the current date use: `CURRENT_DATE`

To get the current time use: `CURRENT_TIME`

To get the current date time: `CURRENT_TIMESTAMP`

Calculating Dates

See documentation sites:

- SQLite
- MS SQL
- PostgreSQL
- MySQL
- Oracle

Formatting Dates

See documentation sites:

- SQLite
- MS SQL
- PostgreSQL
- MySQL
- Oracle