# SQL Cheat Sheet

## Main Syntax

```
SELECT ...
FROM ...
[LEFT/RIGHT/INNER/OUTER] JOIN ... ON ...
WHERE ...
AND/OR ...
GROUP BY ...
ORDER BY ... [ASC/DESC]
LIMIT ...
```

# SELECT

- Choose the columns you want to display (use `*` to select all columns).
- Use functions to calculate things.
- Use `DISTINCT` to eliminate duplicated rows.
- Rename columns using `AS my_name`

### Functions - Operations

- SUM
- COUNT
- AVG
- MIN/MAX

### Functions - Manipulate Data

- DATE_TRUNC
- REGEXP_REPLACE
- COALESCE
- CASE WHEN ...

- LEAST/GREATEST

### Examples

```
SELECT driver_id, driver_name, created_at AS initiated_at
FROM riders
```

# FROM

- State the name of the table from which you want to query the data.

# JOINs

- Think of JOINs as sticking another table to the right of your table stated in the FROM statement.
- Use the ON clause to stick rows from both tables using a key. Rows will be sticked together if this key matches.

### ON

- Indicate the columns from both tables to use as the key

```
SELECT *
FROM table_1
LEFT JOIN table_2 ON table_1.key_1 = table_2.key_2
```

# WHERE, AND/OR

- Only the rows that meet the criteria after this clause will be shown.
- Use parenthesis `()` to evaluate conditions in a specific order

### Examples

- Filtering dates: rows created in January 2020.

```
SELECT *
FROM my_table
WHERE created_at >= '2020-01-01'
AND created_at < '2020-02-01'
```

- Filtering strings:

```
SELECT *
FROM my_table
WHERE driver_name = 'José García' -- exact matching
OR driver_name LIKE 'J%' -- starts with 'J'. Notice the positi
on of the '%'
OR driver_name LIKE '%J' -- ends with 'J'
OR driver_name LIKE '%J%' -- has a 'J' anywhere
```

- Filter + joins

```
SELECT *
FROM table_1
LEFT JOIN table_2 ON table_1.key_1 = table_2 = key_2
WHERE table_1.driver_name = 'José García' -- Notice how we ref
erence a column in a specific table
AND table_2.car_id LIKE '2020%'
```

- Filter like a master: sub-queries.

```
SELECT *
FROM my_table
WHERE driver_id IN (SELECT id FROM drivers WHERE name LIKE '%J
ose%')
```

The sub-query returns a list of all the `driver_id`'s that meet a specific criteria.

# GROUP BY

- Used in conjunction with functions in the SELECT statement
- Name the columns you wish to GROUP BY

## Examples

- Basic use

Example: counts the number of different (driver_id, driver_name) combinations:

```
SELECT driver_id, driver_name, COUNT(*)
FROM rides
GROUP BY driver_id, driver_name
```

- GROUP BY like a pro:

```
SELECT driver_id, driver_name, COUNT(*)
FROM rides
GROUP BY 1, 2
```

Columns are referenced by their position in the query:

- `1` means `driver_id`
- `2` means `driver_name`

# ORDER BY

- Used to sort the data based on a specific column or columns.
- Default order is ASC (ascending)

## Examples

- Basic use

```
SELECT driver_id, driver_name, COUNT(*)
FROM rides
GROUP BY driver_id, driver_name
ORDER BY driver_id
```

- ORDER BY like a pro:

```
SELECT driver_id, driver_name, COUNT(*)
FROM rides
GROUP BY driver_id, driver_name
ORDER BY 3 DESC
```

# Sub-queries

- Sub-queries are tables that you can query from.
- Each table created this way must have an alias name (`AS my_name`)

## Example

```
SELECT driver_id, rides_count - 5 rides_above_threshold
FROM (
  SELECT *
  FROM (
    SELECT driver_id, COUNT(*) rides_count
    FROM rides
    WHERE created_at >= '2020-01-01'
    GROUP BY 1
  ) AS sub_query_1
  WHERE rides_count >= 5
) AS sub_query_2
```

# Common Table Expressions (CTE)

- Similar to sub-queries
- Use when using too many subqueries

## Example

```
WITH
  sub_query_1 AS (
     SELECT driver_id, COUNT(*) rides_count
     FROM rides
     WHERE created_at >= '2020-01-01'
     GROUP BY 1
),
  sub_query_2 AS (
     SELECT *
     FROM sub_query_1
     WHERE rides_count >= 5
)
SELECT driver_id, rides_count - 5 AS rides_above_threshold
FROM sub_query_2
```