Querying Relational Databases Cheatsheet

SQL JOINs

JOINs merge related data from multiple tables together in to result set.

The two most common types of joins are:

- INNER JOIN
- OUTER JOIN

INNER JOINs

INNER JOINs return rows that match from both tables.

```
SELECT <columns> FROM <table 1>
    INNER JOIN <table 2> ON <table 1>.<column> = <table 2>.
```

```
SELECT <columns> FROM <table 1> AS <table 1 alias>
    INNER JOIN <table 2> AS <table 2 alias> ON <table 1 ali
```

Examples:

```
SELECT product_name, category FROM products
    INNER JOIN product_categories ON products.category_id =
SELECT products.product_name, product_categories.category F
```

```
    INNER JOIN product_categories ON products.category_id =
SELECT p.product_name, c.category FROM products AS p
    INNER JOIN product_categories AS c ON p.category_id = c
```

INNER JOINing multiple tables:

```
SELECT <columns> FROM <table 1>
    INNER JOIN <table 2> ON <table 1>.<column> = <table 2>.
    INNER JOIN <table 3> ON <table 1>.<column> = <table 3>.
```

Examples:

```
SELECT users.full_name, sales.amount, products.name FROM sa
        INNER JOIN users ON sales.user_id = users.id
        INNER JOIN products ON sales.product_id = products.
```

OUTER JOINs

There are 3 types of OUTER JOINs:

- LEFT OUTER JOIN - JOINs all matching data and all non-matching rows from the *left* table in the query
- RIGHT OUTER JOIN - JOINs all matching data and all non-matching rows from the *right* table in the query
- FULL OUTER JOIN - JOINs all matching data and then all non-matching rows from both tables.

```
SELECT <columns> FROM <left table>
    LEFT OUTER JOIN <right right> ON <left table>.<column>
```

```
SELECT <columns> FROM <left table> AS <left alias>
    LEFT OUTER JOIN <right table> AS <right alias>
        ON <left alias>.<column> = <right alias>.<column>;
```

Example

If you wanted to get the product count for every category, even categories without products, an `OUTER JOIN` is the best solution. The following two examples will yield the same results, however one is a `LEFT OUTER JOIN` and one is a `RIGHT OUTER JOIN`.

```
SELECT categories.name, COUNT(products.id) AS "Product Coun
    LEFT OUTER JOIN products ON categories.id = products.ca
```

```
SELECT categories.name, COUNT(products.id) AS "Products Cou
    RIGHT OUTER JOIN categories ON categories.id = products
```

Set Operations

Set operations merge data in to one set based on column definitions and the data contained within each column.

The four set operations are:

- UNION
- UNION ALL

- INTERSECT
- EXCEPT

The number of columns need to match. If number of columns don't match it'll result in an error.

```
<query 1> <set operation> <query 2>
SELECT <column> FROM <table 1> <set operation> SELECT <colu
SELECT <column>, <column> FROM <table 1> <set operation> SE
```

UNION Examples

Unions return all distinct values from both data sets with no duplicates.

Get a list of unique restaurants from both north and south malls.

```
SELECT store FROM mall_south WHERE type = "restaurant"
    UNION
SELECT store FROM mall_north WHERE type = "restaurant";
```

Get a list of unique classes taught in two schools. Order them by their class name.

```
SELECT evening_class FROM school_1 UNION SELECT evening_cla
    ORDER BY evening_class ASC;
```

UNION ALL

Union all returns all values from both data sets – with duplicates.

Get a list of all names for boys and girls and order them by name.

```
SELECT boy_name AS name FROM boy_baby_names
    UNION ALL
SELECT girl_name AS name FROM girl_baby_names
    ORDER by name;
```

INTERSECT

Returns only values that are in both data sets.

Get list of classes offered in both schools.

```
SELECT evening_class FROM school_1 INTERSECT SELECT evening
    ORDER BY evening_class ASC;
```

Get list of restaurants at both mall locations.

```
SELECT store FROM mall_south WHERE type = "restaurant"
    INTERSECT
SELECT store FROM mall_north WHERE type = "restaurant";
```

EXCEPT

Returns data from the first data set that's not in the

second.

Get a list of local stores in a mall.

```
SELECT store FROM mall
    EXCEPT
SELECT store FROM all_stores WHERE type = "national"
```

Subqueries

Subqueries are queries within queries. A subquery can also be called an *inner* query with the "parent" query being called the *outer* query.

There are two main ways to use a subquery:

1. In an `IN` condition
2. As a derived or temporary table

A subquery in an `IN` condition must only have one column.

```
SELECT <columns> FROM <table 1> WHERE <table 1>.<column> IN
SELECT <columns> FROM <table 1>
    WHERE <table 1>.<column> IN (SELECT <a single column> F
```

Examples:

Get a list of user's names and emails for users who have spent over 100 dollars in a single transaction.

```
SELECT name, email FROM users
    WHERE id IN (SELECT DISTINCT(user_id) FROM sales WHERE

// OR

SELECT name, email FROM users
    INNER JOIN (SELECT DISTINCT(user_id) FROM sales WHERE s
    ON users.id = best_customers.user_id;
```

Get a list of user's names and emails for users who have spent over 1000 dollars in total.

```
SELECT name, email FROM users WHERE id IN (SELECT user_id F

// OR

SELECT name, email, total FROM users
    INNER JOIN (SELECT user_id, SUM(saleAmount) AS total FR
    ON users.id = ultimate_customers.user_id;
```