Reference card

Querying commands

Data Query Language

Data Query Language (DQL) allows us to communicate with databases, facilitating precise **extraction and manipulation of data**.

SELECT

Select specific or all columns from a specified table. **SELECT*** selects all columns from a table. It is good to use the **LIMIT** statement, especially for large tables.

SELECT ALL

```
SELECT
      *
FROM
      table_name

LIMIT
      100;
```

SELECT columns

```
SELECT
      column1,
      column2,
      column3

FROM
      table_name

LIMIT
      100;
```

SELECT DISTINCT

SELECT DISTINCT eliminates rows where data are **duplicated**. Like **SELECT**, we can check if data in specific column(s) or all columns are duplicated, and remove them.

```
SELECT DISTINCT
      column1,
      column2,
      column3

FROM
      table_name,
```

SELECT WHERE

The **WHERE** clause is used to **filter** records. It is used to extract only those records that fulfil a specified condition.

```
SELECT
      column1,
      column2,
      column3

FROM
      table_name,

WHERE
      column = value;
```

Operators

Operators are used to perform specific operations on one or more operands (values or expressions). Operators are categorised into several types: comparison operators, logical operators, and others like the **LIKE** operator and **WILDCARD**.

Comparison operators

Comparison operators like **greater than** (>), less than (<), **equal to** (=), **not equal to** (<> **or** !=), more than or equal to (<=) are used with **WHERE** to find rows where the outcome of the comparison between a column value and a specified value is **TRUE**.

Greater than

```
SELECT
      *

FROM
      table_name

LIMIT
     column1 > value;
```

Less than

```
SELECT
      *

FROM
      table_name

LIMIT
      column1 < value;
```

Equal to

```
SELECT
      *

FROM
      table_name

WHERE
      column1 = value;
```

Not equal to

```
SELECT
      *

FROM
      table_name

WHERE
      column1 != value;
```

Logical operators

Logical operators are used to **combine or negate conditions**, determining the overall truth of a condition or a set of conditions.

AND

The **AND** operator combines multiple conditions in a query requiring **all** conditions to be met for a row to be included in the result set.

```
SELECT
      column1,
      column2,
      column3

FROM
      table_name

LIMIT
      condition1
      AND condition2;
```

OR

The **OR** operator displays a record in a query where **either** condition separated by the **OR** is **TRUE**.

```
SELECT
      column1,
      column2,
      column3

FROM
      table_name

LIMIT
      condition1
      AND condition2;
```

IN and NOT IN

The **IN** and **NOT IN** operators filter data based on a list of specificed values, allowing for inclusion or exclusion of rows that either do or do not match the provided set of values.

IN

```
SELECT
      column1,
      column2,
      column3

FROM
      table_name

WHERE
      column1

IN
      (
      value1,
      value2,
      value3);
```

NOT IN

```
SELECT
      column1,
      column2,
      column3

FROM
      table_name

WHERE
      column1

NOT IN
      (
      value1,
      value2,
      value3);
```

BETWEEN

The **BETWEEN** operator selects rows within a **specified range**, inclusive of the boundaries, enabling efficient filtering of data based on a given range of values.

BETWEEN

```
SELECT
      column1,
      column2

FROM
      table_name

WHERE
      column1

BETWEEN lower_value AND
      upper_value;
```

NOT BETWEEN

```
SELECT
      column1,
      column2

FROM
      table_name

WHERE
      column1

NOT BETWEEN lower_value AND
      upper_value;
```

LIKE and WILDCARDS

The **LIKE** operator is used with **wildcard characters** such as ('%' and/or '_') to allow for **pattern-based searching**.

% before

```
SELECT
      column1,
      column2

FROM
      table_name

WHERE
      column1

LIKE
      "%XXXX";
```

% after

```
SELECT
      column1,
      column2

FROM
      table_name

WHERE
      column1

LIKE
      "XXXX%";
```

IS NOT NULL/IS NULL

A field with a **NULL** value is a field with **no value**. The **IS NULL** statement will select values in a column that contains **NULL** values. The **IS NOT NULL** statement will select values with no nulls in them.

Select values with IS NOT NULL

```
SELECT
      column1,
      column2

FROM
      table_name

WHERE
      column1 IS NOT NULL;
```

Select values with IS NULL

```
SELECT
      column1,
      column2

FROM
      table_name

WHERE
      column1 IS NULL;
```