How to write efficient queries
Good performance can be achieved by running less queries and optimizing queries.

You should ensure that every query running on Adobe Campaign is optimized to limit the workload on the database and improve user experience.

**Query design**

Efficient queries rely on the use of indexes.

Perform joins using numeric fields instead of string fields.

All joins should use an index.

Avoid performing outer joins. Whenever possible, use the Zero ID record to achieve outer join functionality.

Beware of functions like `Lower(...)`. Remember that when the `Lower` function is used, the Index is not used. Queries using the "like" instruction or the "upper" or "lower" instructions should be examined carefully. The "Upper" should be applied on the user input, not on the database field.

Use the correct data type for joins. Ensure that the "where" clause is the same type as the field. A common mistake is: `iBlacklist='3'` where `iBlacklist` is a numeric field, and '3' signifies a text value. Make sure you know what will be. Full table scans must be avoided, especially for real-time queries, or near real-time queries running every minute.

Use the query's "filtering dimension" feature instead of using the "exists such as" operator

In queries, "exists such as" conditions in filters are not efficient.

The "exists such as" condition is the equivalent of a sub-query in SQL:

```
select iRecipientId from nmsRecipient where iRecipientId IN (select iRecipientId from nmsBroadLog where (...))
```
The best practice is to Use the query’s filtering dimension instead.

The equivalent of the filtering dimension in SQL is the inner join:

```
select iRecipientId from nmsRecipient INNER JOIN nmsBroadLog ON (...)
```

### Join definition in schemas

Defining links on the schema will determine the join conditions.

The linked table should have an unique index on the primary key and the join should be on this field.

Define keys on numeric fields instead of string fields.

- **Build a Dev platform with similar volumes, parameters, and architecture as the Prod platform.**

  Use a TEST (or PrePROD) platform, similar to the PROD environment: same OS, same version, same data, same application, same rights, same volumes whenever possible.

A feature that works in a TEST environment doesn’t mean there will be no issues in a PROD environment where the data may be different. It is important to try to identify the main differences in order to anticipate risks and to prepare solutions.

- **Make configurations that match the target volumes.**

Large volumes require specific configurations. A configuration that worked for 100K recipients may not work for 10M recipients.

Consider how the system will scale when it goes live. Just because something works on a small scale does not mean that it will be
suitable with greater volumes.

Testing should be done with similar volumes to the volume in production. You should also evaluate the effect of changes in volumes (number of calls, size of the database) at peak hours, peak days, and across the life of the project.