**LookML Development Best Practices**

**Naming Convention**

* Follow Looker [naming conventions](https://discourse.looker.com/t/naming-fields-for-readability/712):
  + Name measures with aggregate function or common terms. total\_[FIELD] for sum, count\_[FIELD], avg\_[FIELD], etc.
  + Name ratios descriptively. For example, “Orders Per Purchasing Customers” is clearer than “Orders Percent.”
  + Name yesno fields clearly: “Is Returned” instead of “Returned”
* Avoid the the words date or time in a dimension group because Looker appends each timeframe to the end of the dimension name: created\_date becomes created\_date\_date,created\_date\_month, etc. Simply use created which becomes created\_date, created\_month, etc.  
  [More on timeframes](https://discourse.looker.com/t/timeframes-and-dimension-groups-in-looker/247)

**Project Organization**

* Try to keep one project per connection, with multiple models in each project if they share Views. Use multiple projects per connection when complete isolation of both Models and Views is necessary between git repository and developer teams.
* Review the [include](https://docs.looker.com/reference/model-params/include): statements, remove Views from models if unnecessary or use the wildcard. For example include: “user*.view.lookml” or include: “user*.view” [New LookML] if several user Views (named user\_order\_facts, users, user\_address…) are needed in the model.
* Describe what questions each Explore can answer in the “Documentation” section, using one or many markdown LookML document within each project.

**Model Building**

* Envision entire model from high level. Identify the key subject areas users will want to Explore and select Views to include. Joining many to one from the most granular level typically provides the best query performance.
* Comment out extraneous auto-generated Explores in the Model file (using ‘#’ syntax) to reduce clutter when developing, such as those on dimension tables. Explores can be un-commented later as necessary.
* Use the fewest number of Explores possible that allows users to easily get access to the answers they need. Consider splitting out into different Models for different audiences. The optimal number of Explores is different for every business, however many Explores tend to be confusing for the end user.
* Organize Explores across multiple Models to help the end-user find the correct Explore as easily as possible.
* Limit Access Filter Fields to a different model. Use LookML dashboards to make one dashboard work across several models (i.e. remove any model parameters from dashboard elements).

**Explore Design**

* Limit joins in the Explore to only what is necessary to not overwhelm the user when seeing Views in the field picker.
* Use the fields: parameter for each join or at the Explore level to limit the number of dimensions, measures and filters in the field picker.
* Add a short description to each Explore to specify the purpose and audience using the description: parameter.
* Avoid joining one View into many Explores unless necessary. If the View has fields that scope to other Views this can cause LookML errors and confuse front-end users. Use the fields: parameter to limit this if possible. This also warrants using several View extensions to repeat core fields and add more unique fields for specific Explores.

**Join Design**

* Use sql\_on over foreign\_key for better readability and LookML flexibility.
* Write joins where the base View goes on left to match the relationship parameter, for easy readability.
* Use ${} always. For date joins use the time dimension group option “\_raw” to avoid cast/convert\_tz in join predicates. However, avoid joining on concatenated primary keys declared in Looker - Join on the base fields from the table or persist the primary key for faster queries. More on referencing fields [here](https://discourse.looker.com/t/how-to-reference-views-and-fields-in-lookml/179).
* Always define join relationships in model files using the [relationship:](https://mv4oa2gespowpaagicsxve7mguxh44exjhrwzblzbg6x5fthyrsa.us-east1-c.resources.bumper-boats-00.services.qwiklabs.com/projects/qwiklabs-flights/files/general/ship) parameter. Along with defining primary keys in view files (see View Design), this is necessary to ensure correct aggregates are produced.
* Hard code complex join predicates into map tables (PDTs) to enable faster joins.

**View Design**

* Consider the naming, and familiarity of the end user when writing Views. [See Making Views User Friendly](https://discourse.looker.com/t/making-views-user-friendly/1328).
* Declare a [primary\_key](https://docs.looker.com/reference/field-params/primary_key) in every View on the field that defines unique rows to ensure accuracy. More on why here.
* Use the [hidden:](https://docs.looker.com/reference/field-params/hidden) parameter on dimensions that will never be used to avoid confusion (such as join Keys/IDs and compound primary keys, or those designed solely for application use.)
* Describe fields with the description parameter (description: ‘my great description’). Typical database column names are not descriptive enough for end users.
* Group together types of measures in the View in a consistent manner. This is stylistic but helps others read your model: Grouping by underlying field (counts then sums then averages etc. ) or all measures together by type (counts together, sums together, etc. ) are common methods.
* Use [sets](https://docs.looker.com/reference/view-params/set) to organize groups of fields. This makes it easier to remove fields from Explores and makes the model more maintainable.
* Replace count measures of joined-in tables with count distinct of the foreign key from the Explore, to avoid unnecessary joins (i.e. user\_count = count(distinct orders.user\_id)
* Use [view\_label](https://docs.looker.com/reference/field-params/view_label) and [group\_label](https://docs.looker.com/reference/field-params/group_label) parameters to consolidate dimensions and measures from multiple joined Views that fall under the same category.

**PDT usage**

* Choose the parameter sql\_trigger\_value over persist\_for when data should be ready the first time someone runs an explore or on a schedule. More on why [here](https://discourse.looker.com/t/differences-between-sql-trigger-value-and-persist-for/479).
* Evaluate your sql\_trigger\_value schedules such that tables are not building during business hours/replication processes/peak usage times. Trigger the tables late in the night or early in the morning, after ETL is expected to be completed.
* Always define indexes/distkeys/sortkeys to improve query performance.

**Dashboards Organization/Design**

* Use the [html](https://discourse.looker.com/t/drill-using-a-sparkline-or-other-images/910) or the [link](https://docs.looker.com/reference/field-params/link) parameter in a field to link dashboards (or internal sites) from measures and dimensions
* Single value visualizations are great for highlighting high level KPIs, these do well on the top of the dashboard.

.