**Looker expressions: are used to perform calculations for :**

**Table calculations**

**Custom fields**

**Custom filters**

**\*\*\*\*\*\***

* [Mathematical](https://cloud.google.com/looker/docs/functions-and-operators#math): Number-related functions
* [String](https://cloud.google.com/looker/docs/functions-and-operators#string): Word- and letter-related functions
* [Dates](https://cloud.google.com/looker/docs/functions-and-operators#date): Date- and time-related functions
* [Logical transformation](https://cloud.google.com/looker/docs/functions-and-operators#logical): Includes boolean (true or false) functions and comparison operators
* [Positional transformation](https://cloud.google.com/looker/docs/functions-and-operators#positional): Retrieving values from different rows or pivots

**Field parameters**

bookmark\_border

This page links to all of the LookML parameters that define fields.

There are several types of fields:

* **Dimensions** represent a column in a table, or a computed value based on some sort of column manipulation or combination
* **Dimension Groups** are only used with time-based data, and enable you to create many time-based dimensions at one time
* **Measures** are similar to aggregate functions in SQL (for example, COUNT, SUM, AVG) and represent information about multiple rows
* **Filters** create a filter-only field users can use to provide input to a [templated filter](https://cloud.google.com/looker/docs/templated-filters) or a [conditional join](https://cloud.google.com/looker/docs/reference/param-explore-join-sql-on#conditional_joins)
* **Parameters** create a filter-only field users can use to provide input to a [Liquid {% parameter %} tag](https://cloud.google.com/looker/docs/liquid-variable-reference)

**Measure:**

measure: latest\_record {

type: string

sql: max(${created\_raw}) ;;

}

measure: latest {

type: string or date

sql: min(${created\_raw}) ;;

}

measure: Total\_sale\_price {

type: sum

sql: ${sale\_price} ;;

Every measure will display single value

measure:total\_sales\_500 {

type:yesno

sql: ${Total\_sale\_price}>29000

**Union all :**

select 1 as id,'vinod' as name,100 as profit union all

select 2 as id, 'vino' as name,200 as profit union all

select 3 as id,'vin' as name,300 as profit union all

select 4 as id,'vi' as name,400 as profit union all

select 5 as id,'v' as name,500 as profit

we have to always write explore in models field

measure: previous\_percentage {

type: percent\_of\_previous

sql: ${total\_profit} ;;

}

measure: previous {

type: percent\_of\_previous

sql: ${profit} ;;

}

measure: percent\_total {

type: percent\_of\_total

sql: ${profit} ;;

measure: total\_profit {

type : sum

sql : ${profit}

filters : [name: “shiva”]

measure: count {

type: count

drill\_fields: [detail\*]

**List with date:**

select 1 as id,'vinod' as name,100 as profit,'2023-05-01' as date union all

select 2 as id, 'vino' as name,200 as profit,'2023-05-02' as date union all

select 3 as id,'vin' as name,300 as profit, '2023-05-05'as date union all

select 4 as id,'vi' as name,400 as profit,'2023-04-03' as date union all

select 5 as id,'v' as name,500 as profit, '2023-03-09' as date

models field to write – explore

measure: count {

type: count

filters: [city: "Abbey Hulton, Abbeville,Abbey Green,Abbey Hey"]

drill\_fields: [id, last\_name, first\_name, events.count, order\_items.count]

}

-null is not display 0 values

Measure : age\_count {

Type : count\_distinct

Sql : ${age} ;;

}

Measure: age\_avg{

Type:number

Sql:${total\_age/${age\_count};;

}

Measure: avg

Type: average

Sql:${age} ;;

}

filter: user\_city {

sql: ${city} ;;

}

==================================================