Datasketches Integration

- Sketch functions
  - Naming convention
  - List declared sketch functions
- Integration with materialized views
- BI mode
  - Rewrite COUNT(DISTINCT(X))
  - Rewrite percentile_disc(p) withing group(order by x)
  - Rewrite cume_dist() over (order by id)
  - Rewrite NTILE
  - Rewrite RANK
- Examples
  - Simple distinct counting examples using HLL

Apache DataSketches (https://datasketches.apache.org/) is integrated into Hive via HIVE-22939.
This enables various kind of sketch operations thru regular sql statement.

Sketch functions

Naming convention

All sketch functions are registered using the following naming convention:

ds_{sketchType}_{functionName}

For example we have a function called: **ds_hll_estimate** which could be used to estimate the distinct values from an hll sketch.

sketchType

For detailed info about the sketches themself please refer to the datasketches site!

- frequency
  - hll
  - cpc
  - theta
- frequent items
  - freq
- histograms
  - kll

functionName

name	description
sketch	generates sketch data from input
estimate	computes the estimate for frequency related sketches
union	aggregate function to merge multiple sketches
union_f	unions 2 sketches given in the arguments
n	number of elements
cdf	cumulative distribution
rank	estimates the rank of the given element; returns a value in the range of 0~1
intersect	aggregate to intersect multiple sketches
intersect_f	intersect 2 sketches given in the arguments
stringify	returns the the sketch in a more readable form

List declared sketch functions

Given that we have ~60 functions registered I would recommend to also consider listing/getting info about a single udf.

You could list all functions prefixed by ds_ using:

show functions like 'ds_%';

And you can access the description of a function like:

desc function ds_freq_sketch;

Integration with materialized views

Sketch aggregation(s) are exposed to Calcite by some extensions - which could enable both the usage of an MV in a smaller dimension query; or could help in incremental updates.

BI mode

Usage of sketches can give a performance boost in case we could afford to loose some accuracy. Which could come very handy in case of charts or live dashboards.
The BI mode is about making rewrites automatically to sketch functions if possible.

The BI mode can be enabled using:

```
set hive.optimize.bi.enabled=true;
```

Rewrite COUNT(DISTINCT(X))

This feature can be toggled using the **hive.optimize.bi.rewrite.countdistinct.enabled** conf key

The used distinct sketch family can be configured using: **hive.optimize.bi.rewrite.countdistinct.sketch** (currently only hll is available).

This feature could rewrite

```
select category, count(distinct id) from sketch_input group by category
```

to use a distinct count sketch to answer the query by rewriting it to

```
select category, round(ds_hll_estimate(ds_hll_sketch(id))) from sketch_input
```

Rewrite percentile_disc(p) withing group(order by x)

This feature can be toggled using the **hive.optimize.bi.rewrite.percentile_disc.enabled** conf key

The used histogram sketch family can be configured using: **hive.optimize.bi.rewrite.percentile_disc.sketch** (currently only kll is available).

This feature could rewrite

```
select percentile_disc(0.3) within group(order by id) from sketch_input
```

to use a histogram sketch to answer the query by rewriting to

```
select ds_kll_quantile(ds_kll_sketch(id), 0.3) from sketch_input
```

Rewrite cume_dist() over (order by id)

This feature can be toggled using the **hive.optimize.bi.rewrite.cume_dist.enabled** conf key

The used histogram sketch family can be configured using: hive.optimize.bi.rewrite.cume_dist.sketch (currently only kll is available).

```
select id,cume_dist() over (order by id) from sketch_input
```

to use a histogram sketch to answer the query by rewriting to

```
SELECT id, CAST(DS_KLL_RANK(t2.sketch, idVal) AS DOUBLE)
FROM (SELECT id, CAST(COALESCE(CAST(id AS FLOAT), 340282346638528860000000000000000000000) AS FLOAT) AS idVal
FROM sketch_input) AS t,
(SELECT DS_KLL_SKETCH(CAST(`id` AS FLOAT)) AS `sketch` FROM sketch_input) AS t2
```

Rewrite NTILE

This feature can be toggled using the **hive.optimize.bi.rewrite.ntile.enabled** conf key

The used histogram sketch family can be configured using: hive.optimize.bi.rewrite.ntile.sketch (currently only kll is available).

This feature can rewrite

```
select id,
       ntile(4) over (order by id
from sketch_input
order by id
```

To use a histogram sketch to calculate the NTILE's value:

```
select id,
        case when ceil(ds_kll_cdf(ds, CAST(id AS FLOAT) )[0]*4) < 1 then 1 else ceil(ds_kll_cdf(ds, CAST(id AS
FLOAT) )[0]*4) end
from sketch_input
join ( select ds_kll_sketch(cast(id as float)) as ds from sketch_input ) q
order by id

select id,
                rank() over (order by id),
                case when ds_kll_n(ds) < (ceil(ds_kll_rank(ds, CAST(id AS FLOAT) )*ds_kll_n(ds))+1) then
ds_kll_n(ds) else (ceil(ds_kll_rank(ds, CAST(id AS FLOAT) )*ds_kll_n(ds))+1) end
```

Rewrite RANK

This feature can be toggled using the **hive.optimize.bi.rewrite.rank.enabled** conf key

The used histogram sketch family can be configured using: **hive.optimize.bi.rewrite.rank.sketch** (currently only kll is available).

```
select id,
       rank() over (order by id)
from sketch_input
order by id
```

is rewritten to

```
select id,
       case when ds_kll_n(ds) < (ceil(ds_kll_rank(ds, CAST(id AS FLOAT) )*ds_kll_n(ds))+1) then ds_kll_n(ds)
else (ceil(ds_kll_rank(ds, CAST(id AS FLOAT) )*ds_kll_n(ds))+1) end
from sketch_input
join ( select ds_kll_sketch(cast(id as float)) as ds from sketch_input ) q
order by id
```

Examples

Simple distinct counting examples using HLL

- **Prepare sample table**

```
create table sketch_input (id int, category char(1))
STORED AS ORC
TBLPROPERTIES ('transactional'='true');

insert into table sketch_input values
  (1,'a'),(1, 'a'), (2, 'a'), (3, 'a'), (4, 'a'), (5, 'a'), (6, 'a'), (7, 'a'), (8, 'a'), (9, 'a'), (10,
'a'),
  (6,'b'),(6, 'b'), (7, 'b'), (8, 'b'), (9, 'b'), (10, 'b'), (11, 'b'), (12, 'b'), (13, 'b'), (14, 'b'),
(15, 'b')
;
```

- **Use HLL to compute distinct values using an intermediate table**

```
-- build sketches per category
create temporary table sketch_intermediate (category char(1), sketch binary);
insert into sketch_intermediate select category, ds_hll_sketch(id) from sketch_input group by category;

-- get unique count estimates per category
select category, ds_hll_estimate(sketch) from sketch_intermediate;

-- union sketches across categories and get overall unique count estimate
select ds_hll_estimate(ds_hll_union(sketch)) from sketch_intermediate;
```

- **Use HLL to compute distinct values without intermediate table**

```
select category, ds_hll_estimate(ds_hll_sketch(id)) from sketch_input group by category;
select ds_hll_estimate(ds_hll_sketch(id)) from sketch_input;
```

- **Use HLL to compute distinct values transparently thru BI mode**

```
set hive.optimize.bi.enabled=true;
select category,count(distinct id) from sketch_input group by category;
select count(distinct id) from sketch_input;
```

- **Use HLL to compute distinct values transparently thru BI mode - while utilizing a Materialized View to store the intermediate sketches.**

```
-- create an MV to store precomputed HLL values
create  materialized view mv_1 as
  select category, ds_hll_sketch(id) from sketch_input group by category;

set hive.optimize.bi.enabled=true;
select category,count(distinct id) from sketch_input group by category;
select count(distinct id) from sketch_input;
```