DRUID CHEAT SHEET Cheat Sheet

by kinger1999 via cheatography.com/84184/cs/19885/

Metadata Commands

SELECT * FROM sys.segments

EXPLAIN PLAN FOR <SQL>

INFORMATION_SCHEMA TABLES

SCHEMATA

TABLES

COLUMNS

System Tables

sys.segments

sys.server_segments

sys.tasks

The "sys" schema provides visibility into Druid segments, servers and tasks.

SQL Types

SQL Type	DRUID RUNTIME TYPE
CHAR	STRING
VARCHAR	STRING
DECIMAL	DOUBLE
FLOAT	FLOAT
REAL	DOUBLE
DOUBLE	DOUBLE
BOOLEAN	LONG
TINYINT	LONG
SMALLINT	LONG
INTEGER	LONG
BIGINT	LONG
TIMESTAMP	LONG
DATE	LONG
OTHER	COMPLEX

JDBC CONNECTOR

jdbc:avatica:remote:url=http://BROKER:8-082/druid/v2/sql/avatica/.

You can make Druid SQL queries using the Avatica JDBC driver

PROVIDED BY Imply.io



Aggregation Functions

COUNT(*)

COUNT(DISTINCT expr)

SUM(expr)

MIN(expr)

MAX(expr)

AVG(expr)

Approximate Aggregations

APPROX_COUNT_DISTINCT(expr)

APPROX_COUNT_DISTINCT_DS_HL-L(expr, [lgK, tgtHllType])

APPROX_COUNT_DISTINCT_DS_THETA-(expr, [size])

APPROX_QUANTILE(expr, probability, [resolution])

APPROX_QUANTILE_DS(expr, probability, [k])

APPROX_QUANTILE_FIXED_BUCKETS(-expr, probability, numBuckets, lowerLimit, upperLimit, [outlierHandlingMode])

BLOOM FILTERS

BLOOM_FILTER(expr, numEntries)

BLOOM_FILTER_TEST(<expr>, <serializ-ed-filter>)

COMPARISON OPERATORS

x = y

x <> y

x > y

x >= y

x < y

x <= y

x BETWEEN y AND z

x NOT BETWEEN y AND z

x LIKE pattern [ESCAPE esc]

x NOT LIKE pattern [ESCAPE esc]

x IS NULL

x IS NOT NULL

x IS TRUE

x IS NOT TRUE

x IS FALSE

x IS NOT FALSE

x IN (values)

x NOT IN (values)

x IN (subquery)

x NOT IN (subquery)

x AND y

x OR y

NOT x

By **kinger1999**

cheatography.com/kinger1999/

Published 18th June, 2019.
Last updated 24th June, 2019.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

OTHER FUNCTIONS

CAST(value AS TYPE)

CASE expr WHEN value1 THEN result1 \[WHEN value2 THEN result2 ... \] \[ELSE resultN \] END

CASE WHEN boolean_expr1 THEN result1 \[WHEN boolean_expr2 THEN result2 ... \] \[ELSE resultN \] END

NULLIF(value1, value2)

COALESCE(value1, value2, ...)

NUMERIC FUNCTIONS

ABS(expr)

CEIL(expr)

EXP(expr)

FLOOR(expr)

LN(expr)

LOG10(expr)

POWER(expr, power)

SQRT(expr)

TRUNCATE(expr[, digits])

TRUNC(expr[, digits])

x + y

x - y

x * y

x / y

MOD(x, y)

Numeric functions will return 64 bit integers or 64 bit floats, depending on their inputs.

STRING FUNCTIONS

CONCAT(expr, expr...)

TEXTCAT(expr, expr)

LENGTH(expr)

CHAR_LENGTH(expr)

CHARACTER_LENGTH(expr)

STRLEN(expr)

LOOKUP(expr, lookupName)

LOWER(expr)

POSITION(needle IN haystack [FROM fromIndex])

REGEXP_EXTRACT(expr, pattern, [index])

REPLACE(expr, pattern, replacement)

STRPOS(haystack, needle)

SUBSTRING(expr, index, [length])

SUBSTR(expr, index, [length])

TRIM([BOTH | LEADING | TRAILING] [<chars> FROM] expr)

BTRIM(expr[, chars])

LTRIM(expr[, chars])

UPPER(expr)

String functions accept strings, and return a type appropriate to the function.

TIME FUNCTIONS

CURRENT_TIMESTAMP

CURRENT_DATE

DATE_TRUNC(<unit>, <timestamp_expr>)

TIME_FLOOR(<timestamp_expr>, <period>, [<origin>, [<timezone>]])

TIME_SHIFT(<timestamp_expr>, <period>, <step>, [<timezone>])

TIME_EXTRACT(<timestamp_expr>, [<unit>, [<timezone>]])

TIME_PARSE(<string_expr>, [<pattern>, [<timezone>]])

TIME_FORMAT(<timestamp_expr>, [<pattern>, [<timezone>]])

MILLIS_TO_TIMESTAMP(millis_expr)

TIME FUNCTIONS (cont)

TIMESTAMP_TO_MILLIS(timestamp_expr)

EXTRACT(<unit> FROM timestamp_expr)

FLOOR(timestamp_expr TO <unit>)

CEIL(timestamp_expr TO <unit>)

TIMESTAMPADD(<unit>, <count>, <timestamp>)

timestamp_expr { + | - } <interval_expr>

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com