REDSHIFT – INTRODUCTION

Redshift keywords 1

Amazon Redshift is a distributed database that spreads queries and large volumes of data across multiple workers (nodes) and is part of the AWS ecosystem.

Keyword	Description
columnar database	Optimized on individual columns instead or rows.
PostgreSQL 9 syntax	Is the sql language used for redshift.

Viewing columns and data types

To view columns a data types of a table SVV_REDSHIFT_COLUMNS can be used to see columns and datatypes of internal tables.
SVV_ALL_COLUMNS can be used to see columns and data types of internal and external tables.

Example

```
SELECT column_name, data_type,
character_maxium_lenght,
numeric_precision, numeric_scale

FROM SVV_ALL_COLUMNS

where schema_name =
'spectrumdb' AND table_name =
'ecommerce_sales';
```

Cast()

Same as t-sql cast can be used to change the datatype of a field or value.

Example

```
SELECT
cast(2.00 as INTEGER)
 as our_int
```

Viewing Redshift schemas

To see all databases and the schemas they use the table SVV_REDSHIFT_SCHEMAS is used to see the information from internal tables.
SVV_ALL_SCHEMAS is used to see the information from internal and external tables.

Example

```
SELECT database_name, schema_name, schema_type
FROM SVV_REDSHIFT_SCHEMAS
# database_name| schema_name | schema_type
  dev                | pg_catalog      | local
  production         | pg_catalog      | local
SELECT table_name
FROM SVV_ALL_TABLES
WHERE schma_name = 'spectrumdb'
# table_name
ecommerce_sales
Idaho_site_id
```

Data types

Group	Description	Data types
Numeric	Your average numeric data types.	smallint, integer, bigint, decimal, numeric, double precision, real
Datetime	Your average datetime data types. With timestamp being used for combined date and time.	date, time, timetz, timestamp, timestamptz
Character	Your average string data types.	char, varchar
Boolean	Your Boolean datatype.	boolean
Super	For semi structured data (e.g. jsons) max 16mb.	super
Varbyte	For binary data used for blobs images, videos.	varbyte

Unsupported SQL types

SQL type	Replacement
Datetime	timestamp timestamptz
Serial	integer, bigint
Uuid	varchar
Json	super, varchar
Array	super, varchar
Bit	Boolean, smallint, varchar

REDSHIFT – INTRODUCTION

TO functions

Where cast() needs the for e.g. date already be in the correct format order to not result in an error, the TO functions are leaner during the conversion.

Function	Description
TO_CHAR(type, string, output)	Retrieves the output from string. E.g. month returns the month, ww returns the week number, and day returns the day of the week.
TO_DATE(string, input format)	Converts a string to date field.
TO_NUMBER()	Retreivse the number from a date string.

Example

SELECT TO_CHAR(date '2024-01-14', 'MONTH') AS month_name #JANUARY

SELECT TO_DATE('14-01-2024 02:36:48', 'DD_MM_YYYY') AS our_date; #2024-01-14

SUBSTR and SUBSTRING

Function	Description
SUBSTR(string, int)	Can only be used by the leader node. And returns a substring of a string starting from the index indicated by the int.
SUBSTRING(column, int)	Same as SUBSTR but can be used by the compute node as well.

Example

SELECT *SUBSTR*('datacamp', 4) AS extract;
SELECT *SUBSTRING*(waterusedescription, 5) AS extract

CTE with Redshift

To create an CTE you can use a WITH statement just like in t-sql.

Example

```
WITH division_by_rev AS (
SELECT division_id,
SUM(revenue) as revenue_total
 FROM orders
GROUP BY division_id

SELECT * from division_by_rev
```

Redshift keywords 2 warehouse

Keyword	Description
Leader node	Provides connections, distribute query plans, execute queries, has exclusive functions.
Compute node	Provides data storge, executes code from the leader node on locally stored data.
predicates	Boolean clauses like where, having, on. The leader can push these kinds of requests to compute nodes.
MetaData Catalog	Database component which holds schema info and references a storage location.
Query Engine	Database component plans and executes queries, provides connections.
Storage	Database component hold table data supports multiple file and table formats.
AWS glue data catalog	Stores information about external tables.
AWS s3 bucket	Stores the file that represent the table. E.g. csvs, jsons, text, parquet ect.

Viewing partitions

To view the data partitions between separate nodes the table STV_PARTITIONS can be used.

Example

select host, (used – tossed) / capacity * 100 as percent_used
FROM *STV_PARTITIONS;*

REDSHIFT – INTRODUCTION

Date functions

Many of the t-sql date related functions have equivalents in Redshift.

Function	Description
SYSDATE	Returns the current datetime of the system. (runs on both compute as leader and compute nodes).
GETDATE()	Returns the current datime of the system at the time of start of the statement. (runs on both compute as leader and compute nodes).
CURRENT_ TIME/ NOW	Are equivalents of sysdate and get date but only run the leader node.
TRUNC(timestamp)	Truncates the datetime to just the date part.
DATE_ TRUNC(date part, timestamp)	Truncates the datetime to the specified date part.
DATE_PART(datepart, timestamp)	Extracts the date part from the timestamp. E.g. month, day, year, dayofweek, quarter, timezone.
DATE_CMP(date1, date2)	Returns -1 if date1 is earlier, returns 0 if dates are equal and returns 1 if date 1 later than date2.
DATEDIFF(datepart, date1, date2)	Calculates the differences between date values considering the specified datepart. (Requires the datepart to be part of the date field). Returns a negative value if date2 is earlier than date1.
DATEADD(datepart, int, date)	Adds the int too the datepart of the date if int is negative it will subtract instead.

Example

SELECT *SYSDATE* #2024-01-27 20:05:55.97635

SELECT *TRUNC(GETDATE())* #2024-01-27

SELECT *DATE_TRUNC*('minute', SYSDATE) #2024-01-27 20:05:55

SELECT *DATE_PART*(month, SYSDATE) #1

Window function

Window function lets you run queries and calculations over a moving group of rows in a dataset.

Concept	Description
partitioning	Done through (PARTITION BY) which forms the groups.
Ordering	Done through (ORDE BY) which orders each partition.
Framing	Optional set of restrictions on the rows.

Example

```
SELECT
 division_id,
 sale_date,
revenue,
AVG(revenue
OVER (
 PARTITION BY
deivisio_id,
DATE_PART('year', sale_date),
DATE_PART('month', sale_date))
AS month_avg_revenue
```

Function	Description
LAG	Allows access to any row before current row.
LEAD	Allows access to any row after current row.
RANK()	Same as t-sql rank functions. a unique number per group.

Example

```
SELECT
 division_id,
DATE_PART('year', sale_date) AS sales _year,
DATE_PART('month', sales_date ) AS
sales_month,
COUNT(*) as current_month_sales,
LAG(
COUNT(*),1 )
 OVER (
 PARTITION BY
division_ID
ORDER BY
DATE_PART('year', sale_date),
DATE_PART('month', sale_date))
 AS prior_month_sales
```

REDSHIFT – INTRODUCTION

Transactions

Transactions wrap a series of SQL statement to ensure they all operate as one unit. This enables concurrent operators. (by default every sql statement is a non-grouped transaction) when grouped sysdate returns the same output for each of the grouped transactions. While getdate can return a different output for each statement in grouped transactions.

Creating a grouped transaction

- Opens with a BEGIN or START TRANSACTION function.
- Contains one or more SQL statements (thereby grouping them with a ; after each one.
- Closes with an END or COMMT function.

Example

```
BEGIN;
  query1; #Or stored procedure.
 query2;
END;
```

Table distribution styles

DISTSTYLE	Description	Usage
ALL	Stores the entire table on every node.	Small fact, translation tables that are often needed in joins.
KEY	Distributed by data in the DISTKEY or PRIMARY KEY column.	When we aggregate or join by DISTKEY or PRIMARY KEY.
EVEN	In turn distribution across nodes by row. It spreads the rows across the nodes.	Large tables that don't have keys.
AUTO	Uses ALL style for small tables Key as it grows and has keys or falls back to even.	Is the default.

Keyword	Description
DISTKEY	Generally, the DISKEY is the primary key however if regularly aggregate, join or group on order columns that the primary key we can designate them as the DISTKEY.
SORTKEY	Controls the order the data is stored. Can have multiple separated by comma.

Example

```
CREATE TABLE test
(
   'test_id' INTEGER  PRIMARY KEY,
  'location'  VARCHAR(68),
  'organization_id'  VARCHAR(31) DISTKEY,
  ' organization name' VARCHAR(16)
DISTSTYLE KEY
SORTKEY(location);
```

Table	Description
SVV_TABLE_INFO	Table containing the distyle of the tables.
SVV_REDSHIFT_COLUMNS	Table containing schema, table and their distkey, sortkeys.

REDSHIFT – INTRODUCTION

Importing External table

External tables don't have DISTKEYS or SORTKEYS as they don't support them. Not all table formats support AWS glue (among other iceberg and hive).

Argument	Description
ROW FORMAT	Indicates how the file to be imported is build up. E.g. DELIMITED indicating it has a delimiter.
FIELDS TERMINATED BY	Indicates the delimiter.
STORED AS	How the files are stored in the LOCATION directory E.g. TEXTFILE.
LOCATION	The directory of the files to be loaded in.
TABLE PROPERTIES	Order command that have to be fulfilled while loading in the data. E.g. if the data has headers and if those need to be loaded in.

Example (import a csv)

```
CREATE TABLE dev.test
(  'pizza_id' INTEGER PRIMARY KEY
   'toppings' VARCHAR(64))
ROW FORMAT DELIMITED
FIELD TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 's3://spectrum_id/pizzas_files/'
TABLE PROPERTIES ( 'skip.header.line.count' = '1')
```

Pseudo column	Description
$path	Where is the external file stored.
$size	How many lines does the external file have.

Is_valid_json()

Is_valid_json(json) retursn true if the json provided is valid and readable.

Example

SELECT *IS_VALID_JSON*('{"one":1, "two":2}'); #TRUE

json_extra_array_element

To extract a value from a list using an index JSON_EXTRACT_ARRAY_ELEMENT_TEXT can be used.

Example

SELECT *JSON_EXTRACT_ARRAY_ELEMNT_TEXT*('[1.1,400,13]'), 0); #1.1

Extracting from a json object

To extract specific data from a json object JSON_EXTRACT_PATH_TEXT(json, key) can be used. It accepts two arguments the json and the key you want the value from. the key can also be multiple keys when you want to extract a value from a nested json. The Json needs to be valid.
If the key does not exist it returns null.

Example

SELEC *JSON_EXTRACT_PATH_TEXT* ('{"ONE":1, "two":2}', 'one');
#1

SELECT *JSON_EXTRACT_PATH_TEXT*('{"one_object":{ "nested_three":3, "nested_four":4}, "two":2}', 'one_object', 'nested_three')
#3

REDSHIFT – INTRODUCTION

Casting a super

To cast a json you ::SUPER :: and then the object type your casting from.

Example

```
with location_details as ( select
'{"location": "lisse",}' ::SUPER::
VARCHAR as data #Now accessible
as location_details.data.
```

STL_ALERT_EVENT_LOG table

The STL_ALERT_EVENT_LOG table contains a log from each query that ran and any warnings it might have produced.

Example

```
select * from stl_alert_event_log where query = 1447;
```

EXPLAIN

Running the explain command returns the relative costs and estimating the number of rows to process.

Example

```
EXPLAIN WITH top_ten_division_by_rev AS
(Select division_id, SUM(revenue) AS revenue_total
FROM sales_data
GROUP BY division_id
ORDER BY revenue_total DESC
limit 10 )
```

Column level security table

Security in redshift is by default arranged on column level. To verify you can query the SVV_COLUMN _PRIVILEGES table.
It can be in the form of making the columns un available or masking them using XXXXs.

Example

```
SELECT *
FROM SVV_COLUMN_PREVILEGES
WHERE
 relation_name = 'products';
```

Setting row-level security

Row-level security can be set through the CREATE RLS POLCITY command which prefilters the table on the USING query.

Example

```
CREATE RLS POLICY policy_books
WITH (category VARCHAR(255))
USING (category = 'Dark Academia');
```

Row level security table

To check which row level security policies exists the SVV_RLS_POLICY table can be queried.

Example

```
SELECT
polname  AS policy_name,
polatts AS column_details,
polqual AS condition
FROM SVV_RLS_POLICY;
```

Security log

Admins have another table available to them to see a log when a policy affected a query called SVV_RLS_APPLIED_POLICY.

Example

```
SELECT username, command, relschema,
relname, polname
FROM
SVV_RLS_APPLIED_POLICY;
```