Databricks SQL to BigQuery SQL Migration

Migrating Databricks SQL to BigQuery SQL involves translating Spark SQL dialect (used in Databricks) to BigQuery SQL. This document provides guidance, tools, and Standard Operating Procedures (SOPs) for achieving this migration efficiently while considering the syntactic and functional differences between the two platforms.

Key Considerations

1. **SQL Dialect Differences:** Databricks uses Spark SQL, which differs significantly from BigQuery SQL in syntax and functionality.
2. **Data Handling:** BigQuery operates on a serverless architecture with cost models based on storage and query execution.
3. **Performance Optimization:** Query structures in Databricks may need redesign for BigQuery's execution engine.
4. **Tooling:** Identify tools to automate the translation process wherever possible.

Translation Guidelines

1. Data Types Mapping

Category	Databricks SQL Data Type	BigQuery SQL Data Type	Description
Integer Types	TINYINT	INT64	8-bit integer
	SMALLINT	INT64	16-bit integer
	INT or INTEGER	INT64	32-bit integer
	BIGINT	INT64	64-bit integer
Floating-Point Types	FLOAT or REAL	FLOAT64	64-bit floating point
	DOUBLE	FLOAT64	64-bit floating point (equivalent to BigQuery's FLOAT64)

Decimal/Exact Types	DECIMAL or NUMERIC	NUMERIC or BIGNUMERIC	Fixed-point decimal with user-defined precision and scale. BigQuery has an extended BIGNUMERIC for larger precision.
Boolean Types	BOOLEAN	BOOL	True or False
String Types	STRING or VARCHAR	STRING	Variable-length string
	CHAR	Not Supported	Fixed-length string is not directly supported in BigQuery; use STRING instead.
Date and Time Types	DATE	DATE	Calendar date (year, month, day)
	TIMESTAMP	TIMESTAMP	Timestamp with time zone information
	DATETIME	DATETIME	Timestamp without time zone
Time Interval Types	INTERVAL	Not Supported	Represents a time duration; BigQuery doesn't support this directly.
Binary Types	BINARY	BYTES	Binary data
JSON	JSON	JSON	Semi-structured data type for JSON objects
Array	ARRAY	ARRAY	Ordered list of elements of the same type
Struct	STRUCT	STRUCT	A record containing multiple fields
Map	MAP	Not Supported	Key-value pairs are not natively supported in BigQuery; can use STRUCT or ARRAY of STRUCT as a workaround.

2. SQL Syntax Differences

a) SELECT Queries

Databricks:

Unset

```
SELECT *

FROM sales

WHERE sales_date >= current_date() - INTERVAL 30 DAYS;
```

BigQuery:

Unset

```
SELECT *

FROM `project.dataset.sales`

WHERE sales_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY);
```

b) Window Functions

Databricks:

Unset

```
SELECT emp_id, salary,

    RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank

FROM employees;
```

BigQuery:

Unset

```
SELECT emp_id, salary,
    RANK() OVER (PARTITION BY department ORDER BY salary DESC) AS rank

FROM `project.dataset.employees`;
```

c) Handling NULLs

Databricks:

```
Unset
SELECT COALESCE(column_name, 'default_value')

FROM table_name;
```

BigQuery:

```
Unset
SELECT IFNULL(column_name, 'default_value')

FROM `project.dataset.table_name`;
```

---

3. Common Functions Translation

1. String Functions

Databricks SQL Function	BigQuery SQL Function	Description
UCASE(string)	UPPER(string)	Converts a string to uppercase.
LCASE(string)	LOWER(string)	Converts a string to lowercase.

CONCAT(string1, string2)	CONCAT(string1, string2)	Concatenates two or more strings.
LENGTH(string)	LENGTH(string)	Returns the number of characters in a string.
TRIM(string)	TRIM(string)	Removes leading and trailing spaces from a string.
LTRIM(string)	LTRIM(string)	Removes leading spaces from a string.
RTRIM(string)	RTRIM(string)	Removes trailing spaces from a string.
SUBSTRING(string, pos, len)	SUBSTR(string, pos, len)	Extracts a substring starting at a specified position for a specified length.
REPLACE(string, search, replace)	REPLACE(string, search, replace)	Replaces all occurrences of search with replace in the string.
POSITION(substring IN string)	STRPOS(string, substring)	Returns the position of the first occurrence of a substring.
SPLIT(string, delimiter)	SPLIT(string, delimiter)	Splits a string into an array based on a delimiter.

CONCAT_WS(delim, str1, str2)	ARRAY_TO_STRING(array, delim)	Concatenates strings with a specified delimiter.
REVERSE(string)	REVERSE(string)	Reverses a string.
INITCAP(string)	Not Supported	Converts the first letter of each word in a string to uppercase.

2. Mathematical Functions

Databricks SQL Function	**BigQuery SQL Function**	**Description**
ABS(number)	ABS(number)	Returns the absolute value of a number.
ROUND(number, scale)	ROUND(number, scale)	Rounds a number to the specified number of decimal places.
CEIL(number)	CEIL(number)	Rounds a number up to the nearest integer.
FLOOR(number)	FLOOR(number)	Rounds a number down to the nearest integer.
POWER(base, exponent)	POWER(base, exponent)	Returns the result of raising base to the exponent power.
SQRT(number)	SQRT(number)	Returns the square root of a number.

EXP(number)	EXP(number)	Returns e raised to the power of a number.
LOG10(number)	LOG10(number)	Returns the base-10 logarithm of a number.
LN(number)	LN(number)	Returns the natural logarithm (base e) of a number.
PI()	PI()	Returns the value of pi.
SIN(angle)	SIN(angle)	Returns the sine of an angle in radians.
COS(angle)	COS(angle)	Returns the cosine of an angle in radians.
TAN(angle)	TAN(angle)	Returns the tangent of an angle in radians.
RAND()	RAND()	Generates a random number between 0 and 1.

3. Date and Time Functions

Databricks SQL Function	**BigQuery SQL Function**	**Description**
CURRENT_DATE	CURRENT_DATE()	Returns the current date.
CURRENT_TIMESTAMP	CURRENT_TIMESTAMP()	Returns the current date and time with timezone.

DATEADD(date, interval)	DATE_ADD(date, interval)	Adds a specified interval to a date.
DATEDIFF(date1, date2)	DATE_DIFF(date1, date2, unit)	Returns the difference between two dates in the specified unit (e.g., days, months).
DATE_TRUNC(date, unit)	DATE_TRUNC(date, unit)	Truncates a date to the specified unit (e.g., day, month, year).
FORMAT_DATE(format, date)	FORMAT_DATE(format, date)	Formats a date according to the specified format.
YEAR(date)	EXTRACT(YEAR FROM date)	Extracts the year from a date.
MONTH(date)	EXTRACT(MONTH FROM date)	Extracts the month from a date.
DAY(date)	EXTRACT(DAY FROM date)	Extracts the day from a date.
HOUR(timestamp)	EXTRACT(HOUR FROM timestamp)	Extracts the hour from a timestamp.
MINUTE(timestamp)	EXTRACT(MINUTE FROM timestamp)	Extracts the minute from a timestamp.
SECOND(timestamp)	EXTRACT(SECOND FROM timestamp)	Extracts the second from a timestamp.

4. Conditional Functions

Databricks SQL Function	BigQuery SQL Function	Description
IF(condition, true_value, false_value)	IF(condition, true_value, false_value)	Returns one value if a condition is true and another if false.
CASE WHEN condition THEN value [ELSE value] END	CASE WHEN condition THEN value [ELSE value] END	Conditional branching logic.
COALESCE(value1, value2, ...)	COALESCE(value1, value2, ...)	Returns the first non-NULL value from the list.
NULLIF(value1, value2)	NULLIF(value1, value2)	Returns NULL if two values are equal.

5. Aggregate Functions

Databricks SQL Function	BigQuery SQL Function	Description
COUNT(column)	COUNT(column)	Counts the number of rows or non-NULL values in a column.
SUM(column)	SUM(column)	Calculates the sum of a column's values.
AVG(column)	AVG(column)	Calculates the average of a column's values.

MAX(column)	MAX(column)	Returns the maximum value in a column.
MIN(column)	MIN(column)	Returns the minimum value in a column.
GROUP_CONCAT(column, delim)	STRING_AGG(column, delim)	Concatenates values in a column with a specified delimiter.

6. Window Functions

Databricks SQL Function	**BigQuery SQL Function**	**Description**
ROW_NUMBER()	ROW_NUMBER()	Assigns a unique row number within a window partition.
RANK()	RANK()	Assigns a rank to rows within a window partition, with gaps for ties.
DENSE_RANK()	DENSE_RANK()	Assigns a rank to rows within a window partition without gaps.
LEAD(column, offset)	LEAD(column, offset)	Returns the value of a column at a specified offset ahead in the window.
LAG(column, offset)	LAG(column, offset)	Returns the value of a column at a specified offset behind in the window.

NTILE(n)	NTILE(n)	Divides rows in a window partition into n buckets.

7. H3 Functions

Databricks Function	Bigquery Equivalent Function	Description
h3_boundaryasgeojson(h3CellIdExpr)	`ST_ASGEOJSON(jslibs.h3.ST_H3_BOUNDARY(h3CellIdExpr))`	Returns the polygonal boundary of the input H3 cell in GeoJSON format.
h3_boundaryaswkb(h3CellIdExpr)	`ST_ASBINARY(jslibs.h3.ST_H3_BOUNDARY(h3CellIdExpr))`	Returns the polygonal boundary of the input H3 cell in WKB format.
h3_boundaryaswkt(h3CellIdExpr)	`ST_ASTEXT(jslibs.h3.ST_H3_BOUNDARY(h3CellIdExpr))`	Returns the polygonal boundary of the input H3 cell in WKT format.
h3_centerasgeojson(h3CellIdExpr)	`ST_ASGEOJSON(jslibs.h3.ST_GEOGPOINTFROMH3(h3CellIdExpr))`	Returns the center of the input H3 cell as a point in GeoJSON format.
h3_centeraswkb(h3CellIdExpr)	`ST_ASBINARY(jslibs.h3.ST_GEOGPOINTFROMH3(h3CellIdExpr))`	Returns the center of the input H3 cell as a point in WKB format.
h3_centeraswkt(h3CellIdExpr)	`ST_ASTEXT(jslibs.h3.ST_GEOGPOINTFROMH3(h3CellIdExpr))`	Returns the center of the input H3 cell as a point in WKT format.
h3_compact(h3CellIdsExpr)	`jslibs.h3.compact(h3CellIdExpr)`	Compacts the input set of H3 cell IDs as best as possible.
h3_coverash3(geographyExpr, resolutionExpr)	`jslibs.h3.ST_H3_POLYFILLFROMGEOG(geographyExpr, resolutionExpr)`	Returns an ARRAY of H3 cell IDs (represented as BIGINT) corresponding to the minimal set of hexagons or pentagons, of the specified resolution, that fully cover the input linear or areal geography.

h3_coverash3string(geographyExpr, resolutionExpr)	`CREATE TEMP FUNCTION h3_coverash3string(geographyExpr GEOGRAPHY, resolutionExpr INT64) AS ( ARRAY_AGG(CAST(h3_index AS STRING)) FROM ( SELECT h3_index FROM UNNEST(jslibs.h3.ST_H3_POLYFILLFROMGEOG(geographyExpr, resolutionExpr)) AS h3_index ) );`	Returns an ARRAY of H3 cell IDs (represented as STRING) corresponding to the minimal set of hexagons or pentagons, of the specified resolution, that fully cover the input linear or areal geography.
h3_distance(h3CellId1Expr, h3CellId2Expr)	No direct equivalent Inaccurate function might be: `ST_DISTANCE( jslibs.h3.ST_GEOGPOINTFROMH3(h3CellId1), jslibs.h3.ST_GEOGPOINTFROMH3(h3CellId2) );`	Returns the grid distance of the two input H3 cell IDs.
h3_h3tostring(h3CellIdExpr)	`CAST(h3CellIdExpr AS STRING)`	Converts the input H3 cell ID to its equivalent hexadecimal string representation.
h3_hexring(h3CellIdExpr, kExpr)	jslibs`.h3.hexRing(h3CellIdExpr, KExpr)`	Returns an array of H3 cell IDs that form a hollow hexagonal ring centered at the origin H3 cell and that are at grid distance k from the origin H3 cell.
h3_ischildof(h3CellId1Expr, h3CellId2Expr)	No direct equivalent	Returns true if the first H3 cell ID is equal to or a child of the second H3 cell ID.
h3_ispentagon(h3CellIdExpr)	`CREATE TEMP FUNCTION h3_ispentagon(h3CellIdExpr INT64) AS ( CASE WHEN ARRAY_LENGTH(ST_GEOGPOINTS(ST_H3_BOUNDARY(h3CellIdExpr))) = 5 THEN TRUE -- Pentagons have 5 vertices ELSE FALSE -- Otherwise, it's a hexagon END );`	Returns true if the input BIGINT or hexadecimal STRING corresponds to a pentagonal H3 cell or not.

h3_isvalid(expr)	`jslibs.h3.h3IsValid(h3CellExpr)`	Returns true if the input BIGINT or STRING is a valid H3 cell ID.
h3_kring(h3CellIdExpr, kExpr)	`jslibs.h3.kRing(h3Index, ringSize)`	Returns the H3 cell IDs that are within (grid) distance `k` of the origin cell ID.
h3_kringdistances(h3CellIdExpr, kExpr)	No direct equivalent	Returns all H3 cell IDs (represented as long integers or strings) within grid distance `k` from the origin H3 cell ID, along with their distance from the origin H3 cell ID.
h3_longlatash3(longitudeExpr, latitudeExpr, resolutionExpr)	`jslibs.h3.ST_H3(ST_GEOGPOINT(longitudeExpr, latitudeExpr), resolutionExpr)`	Returns the H3 cell ID (as a BIGINT) corresponding to the provided longitude and latitude at the specified resolution.

h3_longlatash3string(longitudeExpr, latitudeExpr, resolutionExpr)	`CREATE TEMP FUNCTION h3_longlatash3string(longitudeExpr FLOAT64, latitudeExpr FLOAT64, resolutionExpr INT64) AS (`
	
	`CAST(jslibs.h3.ST_H3(ST_GEOGPOINT(longitudeExpr, latitudeExpr), resolutionExpr) AS STRING)`
	`);`
		Returns the H3 cell ID (as a hexadecimal STRING) corresponding to the provided longitude and latitude at the specified resolution.

h3_maxchild(h3CellIdExpr, resolutionExpr)	No direct equivalent	Returns the child of maximum value of the input H3 cell at the specified resolution.
h3_minchild(h3CellIdExpr, resolutionExpr)	No direct equivalent	Returns the child of minimum value of the input H3 cell at the specified resolution.
h3_pointash3(geographyExpr, resolutionExpr)	`jslibs.h3.ST_H3(geographyExpr, resolutionExpr)`	Returns the H3 cell ID (as a BIGINT) corresponding to the provided point at the specified resolution.

h3_pointash3string(geographyExpr, resolutionExpr)	No direct equivalent, but achievable	Returns the H3 cell ID (as a STRING) corresponding to the provided point at the specified resolution.
h3_polyfillash3(geographyExpr, resolutionExpr)	`jslibs.h3.ST_H3_POLYFILLFROMGEOG(geographyExpr, resolutionExpr)`	Returns an ARRAY of H3 cell IDs (represented as BIGINT) corresponding to hexagons or pentagons, of the specified resolution, that are contained by the input areal geography.
h3_polyfillash3string(geographyExpr, resolutionExpr)	No direct equivalent, but achievable	Returns an ARRAY of H3 cell IDs (represented as STRING) corresponding to hexagons or pentagons, of the specified resolution, that are contained by the input areal geography.
h3_resolution(h3CellIdExpr)	`jslibs.h3.h3GetResolution(h3CellIdExpr)`	Returns the resolution of the input H3 cell ID.
h3_stringtoh3(h3CellIdStringExpr)	`CREATE TEMP FUNCTION h3_stringtoh3(h3CellIdStringExpr STRING) AS ( CAST(h3CellIdStringExpr AS INT64) );`	Converts the input string, which is expected to be a hexadecimal string representing an H3 cell ID, to the corresponding BIGINT representation of the H3 cell ID.
h3_tessellateaswkb(geographyExpr, resolutionExpr)	No direct equivalent	Returns a tessellation of the input geography using H3 cells at the specified resolution.
h3_tochildren(h3CellIdExpr, resolutionExpr)	`jslibs.h3.h3ToChildren(h3Index STRING, resolution NUMERIC)`	Returns an array of the children H3 cell IDs of the input H3 cell ID at the specified resolution.
h3_toparent(h3CellIdExpr, resolutionExpr)	`jslibs.h3.h3ToParent(h3Index STRING, resolution NUMERIC)`	Returns the parent H3 cell ID of the input H3 cell ID at the specified resolution.
h3_try_distance(h3CellId1Expr, h3CellId2Expr)	No direct equivalent	Returns the grid distance of the two input H3 cell IDs of the same resolution, or NULL if the distance is undefined.

h3_try_polyfillash3(geographyExpr, resolutionExpr)	`jslibs.h3.ST_H3_POLYFILLFROMGEOG(geographyExpr, resolutionExpr)`	Returns an ARRAY of H3 cell IDs (represented as BIGINT) corresponding to hexagons or pentagons, of the specified resolution, that are contained by the input areal geography.
h3_try_polyfillash3string(geographyExpr, resolutionExpr)	No direct equivalent, but achievable	Returns an ARRAY of H3 cell IDs (represented as STRING) corresponding to hexagons or pentagons, of the specified resolution, that are contained by the input areal geography.

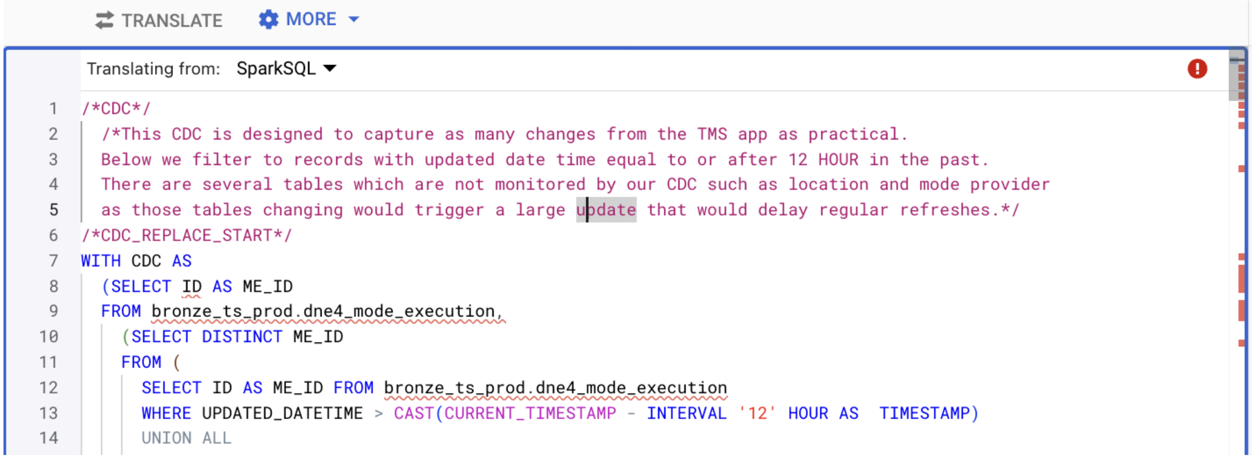
h3_try_validate(h3CellIdExpr)	`CREATE TEMP FUNCTION h3_try_validate(h3CellId STRING)`
	`RETURNS STRING`
	`LANGUAGE js AS """`
	`  return h3.h3IsValid(h3CellId) ? h3CellId : null;`
	`"""`
	`OPTIONS (`
	`  library=["gs://bigquery-jslibs/h3-js.umd.js"]`
	`);`
		Returns the input value, that is of type BIGINT or STRING, if it corresponds to a valid H3 cell ID, or NULL otherwise.

h3_uncompact(h3CellIdsExpr, resolutionExpr)	`jslibs.h3.uncompact(h3CellIdExpr, resolutionExpr)`	Uncompacts the input set of H3 cell IDs to the specified resolution.
h3_validate(h3CellIdExpr)	No direct equivalent	Returns the input value, that is of type BIGINT or STRING, if it corresponds to a valid H3 cell ID, or emits an error otherwise.

Translation Tools

1. Bigquery Translation Service:

Databricks SQL has its own functions and syntax, but it is fundamentally Spark SQL. To translate with Bigquery Translation Service for databricks SQL, you may select spark SQL as the source dialect for translation.



Translating from: SparkSQL

```
/*CDC*/
  /*This CDC is designed to capture as many changes from the TMS app as practical.
  Below we filter to records with updated date time equal to or after 12 HOUR in the past.
  There are several tables which are not monitored by our CDC such as location and mode provider
  as those tables changing would trigger a large update that would delay regular refreshes.*/
/*CDC_REPLACE_START*/
WITH CDC AS
  (SELECT ID AS ME_ID
  FROM bronze_ts_prod.dne4_mode_execution,
    (SELECT DISTINCT ME_ID
    FROM (
      SELECT ID AS ME_ID FROM bronze_ts_prod.dne4_mode_execution
      WHERE UPDATED_DATETIME > CAST(CURRENT_TIMESTAMP - INTERVAL '12' HOUR AS  TIMESTAMP)
      UNION ALL
```

Cons

1. Spark SQL is not detected in cases such as below:
    a. Use of date intervals in plural such as DAYS, HOURS, SECONDS, etc. To proceed with translation manually replace this to singular form such as DAY, HOUR, SECOND, etc.
    b. Use of IS TRUE. Change to = TRUE for translation.
2. This tool removes comments from the SQL code.
3. Convert all table names to lowercase.

2. Python Libraries

https://github.com/tobymao/sqlglot

Cons

1. Requires smaller lines of code to work with.
2. Development efforts to get desired functionality.

3.[Recommended] Gemini

Leverage Gemini APIs to translate the given SQL query into a functionally equivalent BigQuery query. Furnish detailed translation guidelines that address any differences in syntax, data types, or built-in functions between the source SQL dialect and BigQuery SQL. Clearly outline the expected results and behavior of the translated query, including any potential performance considerations.

References

- Data types | BigQuery | Google Cloud
- https://github.com/uber/h3-js
- https://github.com/dtws/bigquery-jslibs/tree/master
- H3 geospatial functions | Databricks on AWS