DATABRICKS CHEATSHEET

Databricks SQL (DB SQL) is a serverless data warehouse on the Databricks Lakehouse Platform that lets you run all your SQL and BI applications at scale with up to 12x better price/performance, a unified governance model, open formats and APIs, and your tools of choice – no lock-in.

CREATE TABLES

CREATE TABLE

```
--Create a table and define its schema.
CREATE TABLE default.sales (
   transaction_datetime TIMESTAMP,
   refund_datetime TIMESTAMP,
   bank_zip INT,
   customer_zip INT
);
```

CREATE VIEW

```
CREATE VIEW mytempview
AS SELECT * FROM default.sales;
```

CREATE OR REPLACE TABLE

```
CREATE OR REPLACE TABLE default.sales
parquet.`/path/to/data`;
```

ALTER TABLE

RENAME TABLE

```
ALTER TABLE sales
RENAME TO salesperson;
```

RENAME COLUMN

```
ALTER TABLE sales
RENAME COLUMN customer_first_name TO customer_name;
```

ADD COLUMNS

```
ALTER TABLE sales ADD columns (time TIMESTAMP, col_name1
data_type2);
```

CHECK (CONSTRAINTS)

```
--Add a CHECK constraint
ALTER TABLE sales
ADD CONSTRAINT dateWithinRange CHECK (year > '2000-01-
01');
```

NOT NULL (CONSTRAINTS)

```
--Add a NOT NULL constraint
ALTER TABLE sales
ADD CONSTRAINT customer_name IS NOT NULL;
```

DROP CONSTRAINT (CONSTRAINTS)

```
ALTER TABLE default.sales
DROP CONSTRAINT dateWithinRange;
```

DELETE / DROP A TABLE

DELETE

```
--Delete rows in a table based upon a condition
DELETE FROM sales
WHERE predicate;
```

DROP TABLE

```
DROP TABLE [IF EXISTS] sales;
```

TRUNCATE

```
--Keep a table but delete all of its data.
TRUNCATE TABLE sales;
```

ADD/MODIFY DATA

UPDATE

```
--Update column values for rows that match a predicate
UPDATE sales
SET bank_office = 'Augusta'
WHERE employee_state = 'Maine';
```

INSERT INTO

```
--Insert comma separated values directly into a table.
INSERT [OVERWRITE] INTO mytable VALUES
('Harper Bryant', 'Employee', 98101),
('Sara Brown', 'Contractor', 48103);
```

MERGE INTO

```
--Upsert (update + insert) using MERGE
MERGE INTO target
USING updates
ON target.Id = updates.Id
WHEN MATCHED AND target.delete_flag = "true" THEN
   DELETE
WHEN MATCHED THEN
   UPDATE SET *
WHEN NOT MATCHED THEN
   INSERT (date, Id, data) -- or, use INSERT *
   VALUES (date, Id, data);
```

IDENTITY COLUMNS

AUTO-INCREMENTING IDENTITY COLUMNS

```
--Add an auto-incrementing identity column
CREATE TABLE sales
(id BIGINT GENERATED ALWAYS AS IDENITY COMMENT 'Surrogate
key for AccountID',
accountid BIGINT,
samplecolumn STRING
);
```

SHOW IDENTITY COLUMNS

```
--Returns the CREATE TABLE statement that was used to
create a given table or view. Allows you to see which
column(s) are identity columns.
SHOW CREATE TABLE sales;
```

JOINS

JOIN

```
--Join two tables (via inner, outer, left, or right join)
SELECT city.name, country.name
FROM city
[INNER|OUTER|LEFT|RIGHT] JOIN country
  ON city.country_id = country.id;
```

COMMON SELECT QUERIES

SUBQUERIES

```
--Query an intermediate result set using a subquery.
SELECT * FROM sales
WHERE sales_id IN (
  SELECT DISTINCT sales_id
  FROM visit
);
```

ALIAS COLUMN

```
--Alias a column
SELECT sales_id AS sales_id_new
FROM sales;
```

ALIAS TABLE

```
--Alias a table
SELECT * FROM my_sales AS m;
```

ORDER BY

```
--Return a table sorted by a column's values. Values
returned in ascending order by default, or specify DESC.
SELECT productname, sales_id FROM sales
ORDER BY sales_id [DESC];
```

WHERE

```
--Filter a table based upon rows that match one or more
specific predicates (text or numeric filtering)
SELECT * FROM sales
WHERE product_name = "Lego set" AND sales_id > 50000;
```

JSON

```
--extract values from a JSON string using the : operator,
delimeters and identifiers
SELECT raw:owner, raw:OWNER, raw:['owner'], raw:['OWNER']
FROM sales;

--Extract nested fields from JSON string using the
: operator and dot notation
SELECT raw:store.bicycle FROM sales;

--Extract values from an array in JSON using the
: operator
SELECT raw:store.fruit[0], raw:store.fruit[1] FROM sales;
```

CLONE

```
-- Deep clone is a complete, independent copy of the source
table
CREATE OR REPLACE TABLE default.sales DEEP CLONE
parquet.`/path/to/data`;

-- Shallow clone is a copy of the source table’s definition,
but refers to the source table’s files
CREATE OR REPLACE TABLE default.sales SHALLOW CLONE
parquet.`/path/to/data`;
```

DBSQL CHEATSHEET

COMMON AGGREGATIONS

COUNT

```
--View count of distinct records in a table
SELECT COUNT([DISTINCT] sales)
FROM orderhistory;
```

AVERAGE/MIN/MAX

```
--View average (mean), sum, or min and max values in a
column
SELECT AVG(sales), SUM(sales), MIN(sales), MAX(sales)
FROM orderhistory;
```

GROUP BY/HAVING

```
--View an aggregation grouped by a column's values.
Optionally, specify a predicate using the HAVING clause
that rows must match to be included in the aggregation.
SELECT SUM(sales)
FROM orderhistory
GROUP BY country
[HAVING item_type="soup"];
```

PERMISSIONS

GRANT

```
-- Grant database and table permissions for admin group
GRANT ALL PRIVILEGES ON [DATABASE default|TABLE sales] TO
`name@email.com`| admins;
```

REVOKE

```
--Revoke privileges on databases or tables
REVOKE [SELECT TABLE|ALL PRIVILEGES|CREATE TABLE|etc.] ON
sales FROM [`name@email.com`|admins];
```

SHOW GRANT

```
--Show a user's permissions on a table
SHOW GRANT `user@example.com` ON TABLE default.sales;
```

INFORMATION SCHEMA

INFORMATION SCHEMA

```
--View all tables that have been created in the last 24
hours
SELECT table_name, table_owner, created_by, last_altered,
last_altered_by, table_catalog
  FROM system.information_schema.tables
  WHERE datediff(now(), last_altered) < 1;

--View how many tables you have in each schema
SELECT table_schema, count(table_name)
  FROM system.information_schema.tables
  WHERE table_schema = 'tpch'
  GROUP BY table_schema
  ORDER BY 2 DESC
```

USE

```
--Switch to a different database; the database default is
used if none is specified.
USE database_name;
```

DELTA LAKE

CHANGE DATA FEED

```
--Read table changes starting at a specified version number
SELECT * FROM table_changes('sales', <start version #>)
--Enable Change Data Feed on Delta Lake table
ALTER TABLE sales SET TBLPROPERTIES
(delta.enableChangeDataFeed = true);
```

CONVERT TO DELTA

```
--Convert a table to Delta Lake format
CONVERT TO DELTA sales;
```

VACUUM

```
--Delete files no longer used by the table from cloud
storage
VACUUM sales [RETAIN num HOURS] [DRY RUN];
```

TIME TRAVEL

```
--Query historical versions of a Delta Lake table by
version number or timestamp
SELECT * FROM table_name [VERSION AS OF 0 | TIMESTAMP AS
OF "2020-12-18"]
--View Delta Lake transaction log (table history)
DESCRIBE HISTORY sales;
```

DESCRIBE

```
--View [detailed] information about a database or table
DESCRIBE [DETAIL] sales;
```

GEOSPATIAL FUNCTIONS

H3

```
--Returns the H3 cell ID (as a BIGINT) corresponding to the
provided longitude and latitude at the specified resolution
SELECT h3_longlatash3(longitudeExpr, latitudeExpr,
resolutionExpr)

--Returns an ARRAY of H3 cell IDs (represented as a BIGINTs)
corresponding to hexagons or pentagons, of the specified
resolution, that are contained by the input areal geography
SELECT h3_polyfillash3(geographyExpr, resolutionExpr)

--Returns the H3 cell IDs that are within (grid) distance k
of the origin cell ID
SELECT 3_kring(h3CellIdExpr, kExpr)
--Returns the grid distance of the two input H3 cell IDs
SELECT h3_distance(h3CellId1Expr, h3CellId2Expr)
--Returns the parent H3 cell ID of the input H3 cell ID at
the specified resolution
SELECT h3_toparent(h3CellIdExpr, resolutionExpr)
```

CTE

CTE

```
--Create a common table expression (CTE) that can be
easily reused in other queries.
WITH common_table_expression_name
AS (
  SELECT
    product_name as product,
    AVG(sales) as avg_sales
  FROM orderhistory
  GROUP BY product
)
SELECT * FROM common_table_expression_name
```

PERFORMANCE TUNING

CACHE

```
--Cache a table in memory to speed up queries.
CACHE SELECT sales;
```

EXPLAIN

```
--View the physical plan for execution of a given SQL
statement.
EXPLAIN [EXTENDED] SELECT * FROM sales;
```

TUNE WIDE TABLES

```
 --Sets the number of columns to collect statistics on
 ALTER TABLE SET TBLPROPERTIES
 ('delta.dataSkippingNumIndexedCols' = 64);
```

OPTIMIZE

```
--OPTIMIZE Delta tables, bin packs tables for better
performance
OPTIMIZE sales
```

ANALYZE

```
--Analyze table to collect statistics on entire column
ANALYZE TABLE sales COMPUTE STATISTICS FOR ALL COLUMNS;
```

OPTIMIZE/ZORDER

```
--Periodic OPTIMIZE and ZORDER, run on a nightly basis
OPTIMIZE customer_table ZORDER BY customer_id, customer_seq;
```

DATA INGESTION

COPY INTO

```
COPY INTO iot_devices
FROM "/databricks-datasets/iot/"
FILEFORMAT = JSON|CSV|PARQUET|etc.;
```

CREATE FUNCTION

CREATE FUNCTION

```
-- Create a permanent function with parameters.
CREATE FUNCTION area(x DOUBLE, y DOUBLE) RETURNS DOUBLE
RETURN x * y;

-- Use a SQL function in the SELECT clause of a query.
SELECT area(c1, c2) AS area FROM t;

-- Use a SQL function in the WHERE clause of a query.
SELECT * FROM t WHERE area(c1, c2) > 0;

-- Compose SQL functions.
CREATE FUNCTION square(x DOUBLE) RETURNS DOUBLE RETURN
area(x, x);
SELECT c1, square(c1) AS square FROM t

- Create a non-deterministic function
CREATE FUNCTION roll_dice()
RETURNS INT
NOT DETERMINISTIC
CONTAINS SQL
COMMENT 'Roll a single 6 sided die'
RETURN (rand() * 6)::INT + 1;

-- Roll a single 6-sided die
SELECT roll_dice();
```