databricks

SQL CHEAT SHEET

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.

View Databricks SQL Demos

Learn More About Databricks SQL

Ready to get started?

CREATE TABLES

CREATE TABLE

```
--Create a table and define its schema.
CREATE TABLE default.nyctaxi_trips (
   pickup_datetime TIMESTAMP,
   dropoff_datetime TIMESTAMP,
   trip_distance DOUBLE,
   pickup_zip INT,
   dropoff_zip INT
);
```

VIEW

```
--Create temporary view
CREATE TEMPORARY VIEW mytempview
AS SELECT * FROM dbname.tablename;
```

ALTER TABLE

RENAME TABLE

```
--Rename a table
ALTER TABLE table_name
RENAME TO new_table_name;
```

RENAME COLUMN

```
--Rename a column
ALTER TABLE my_table
RENAME COLUMN original_column_name TO new_column_name;
```

ADD COLUMNS

```
--Add new columns to a Delta Lake table
ALTER TABLE table_name ADD COLUMNS (col_name1 data_type1,
col_name1 data_type2);
```

CHECK (CONSTRAINTS)

```
--Add a CHECK constraint
ALTER TABLE default.people10m
ADD CONSTRAINT dateWithinRange CHECK (birthDate > '1900-
01-01');
```

NOT NULL (CONSTRAINTS)

```
--Add a NOT NULL constraint
ALTER TABLE table_name
ADD CONSTRAINT column_a IS NOT NULL;
```

DROP CONSTRAINT (CONSTRAINTS)

```
--Drop a constraint
ALTER TABLE default.people10m
DROP CONSTRAINT dateWithinRange;
```

ADD/MODIFY DATA

UPDATE

```
--Update column values for rows that match a predicate
UPDATE employee_table
SET home_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);
```

COPY INTO

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

DELETE / DROP A TABLE

DELETE

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

DROP TABLE

```
--Drop a table
DROP TABLE [IF EXISTS] table_name;
```

TRUNCATE

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

IDENTITY COLUMNS

IDENTITY COLUMNS

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

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 mytable;
```

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;
```

DATABASES

USE

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

COMMON SELECT QUERIES

SUBQUERIES

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

ALIAS COLUMN

```
--Alias a column
SELECT dev_id_capture_4 AS device_id
FROM mytable;
```

ALIAS TABLE

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

SELECT

```
--Query from database and table (fully qualifying the
table)
SELECT * FROM catalogname.databasename.tablename;
```

SELECT

```
--Select specific columns from table
SELECT accountid, devicetype FROM devices;
```

ORDER BY

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

WHERE

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

databricks

databricks SQL CHEAT SHEET P2

COMMON AGGREGATIONS

COUNT

```
--View count of records in a table, or a count of
[distinct] records in a table
SELECT COUNT(*)|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"];
```

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
);
```

DATA INGESTION

COPY INTO

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

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('my_table', <start version #>)

--Enable Change Data Feed on Delta Lake table
ALTER TABLE my_table SET TBLPROPERTIES
(delta.enableChangeDataFeed = true);
```

CONVERT TO DELTA

```
--Convert a table to Delta Lake format
CONVERT TO DELTA [dbName.]tableName;
```

VACUUM

```
--Delete files no longer used by the table from cloud
storage
VACUUM table_name [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 mytable;
```

DESCRIBE

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

PERFORMANCE TUNING

CACHE

```
--Cache a table in memory to speed up queries.
CACHE TABLE tablename;
```

EXPLAIN

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

AUTO-TUNE

```
--Use Auto-Tune for File Sizes
ALTER TABLE SET TBLPROPERTIES
('delta.tuneFileSizesForRewrites', True);
```

OPTIMIZE

```
--OPTIMIZE Delta tables, and Z-Order by selective join
keys or common selective query predicates
OPTIMIZE mytable ZORDER BY joinkey1, predicate2;
```

ANALYZE

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

NULL SEMANTICS

```
--comment here about Null semantics
```

PERMISSIONS

GRANT

```
--View count of records in a table, or a count of
[distinct] records in a table
GRANT ALL PRIVILEGES ON [DATABASE mydatabase|TABLE mytable]
TO `name@email.com`| GROUPNAME;
```

REVOKE

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

SHOW GRANT

```
--Show a user's permissions on a table
SHOW GRANT `user@example.com` ON TABLE default.people10m;
```