Standard SQL Cheat Sheet

Standard SQL (ISO/IEC 9075) only · AI-generated summary — verify against your DBMS docs · see the full SQL Reference for details

DDL — DEFINE

CREATE TABLE

```
CREATE TABLE t (id INT PRIMARY KEY,
  name VARCHAR(50) NOT NULL);
```

ALTER TABLE

```
ALTER TABLE t ADD COLUMN age INT;
ALTER TABLE t DROP COLUMN age;
```

DROP / CREATE VIEW / INDEX

```
DROP TABLE t CASCADE;
CREATE VIEW v AS SELECT * FROM t;
CREATE INDEX idx ON t(name);
```

Constraints

```
PRIMARY KEY, FOREIGN KEY REFERENCES,
UNIQUE, NOT NULL, CHECK (x>0), DEFAULT
```

SELECT — QUERY

Clause order

```
SELECT DISTINCT a, COUNT(*) FROM t
WHERE a IS NOT NULL
GROUP BY a HAVING COUNT(*) > 1
ORDER BY a
OFFSET 0 ROWS FETCH FIRST 10 ROWS ONLY;
```

WHERE predicates

```
x BETWEEN 1 AND 10, x IN (1,2), x LIKE 'a%',
x IS NULL, a AND/OR/NOT b
```

Joins

```
FROM a JOIN b ON a.id = b.id
LEFT/RIGHT/FULL OUTER JOIN, CROSS JOIN
```

DML — MODIFY

INSERT

```
INSERT INTO t (id, name) VALUES (1,'A');
INSERT INTO t SELECT * FROM staging;
```

UPDATE / DELETE

```
UPDATE t SET name='B' WHERE id=1;
DELETE FROM t WHERE id=1;
```

MERGE / TRUNCATE

```
MERGE INTO t USING s ON t.id=s.id
  WHEN MATCHED THEN UPDATE SET ...
  WHEN NOT MATCHED THEN INSERT ...;
TRUNCATE TABLE t;
```

TRANSACTIONS

Control

```
START TRANSACTION;
SAVEPOINT sp1;
ROLLBACK TO SAVEPOINT sp1;
COMMIT; / ROLLBACK;
```

Isolation levels

```
SET TRANSACTION ISOLATION LEVEL
  READ COMMITTED | REPEATABLE READ |
  SERIALIZABLE | READ UNCOMMITTED;
```

DCL — ACCESS

GRANT / REVOKE

```
GRANT SELECT, INSERT ON t TO role1
  WITH GRANT OPTION;
REVOKE INSERT ON t FROM role1;
```

SET

```
SET ROLE role1;
SET SESSION CHARACTERISTICS AS
  TRANSACTION READ ONLY;
```

COMMON FUNCTIONS

String / Numeric

```
SUBSTRING(s FROM 1 FOR 3), TRIM(s),
UPPER(s), CONCAT(a,b), ABS(n), ROUND(n,2)
```

Math / Trig

```
SQRT(n), POWER(n,2), EXP(n), LN(n), LOG10(n)
SIN(x), COS(x), TAN(x), ASIN/ACOS/ATAN(x)
```

Date / Cast

```
CURRENT_DATE, CURRENT_TIMESTAMP,
LOCALTIME, LOCALTIMESTAMP,
EXTRACT(YEAR FROM d), CAST(x AS INT)
```

AGGREGATE & WINDOW

Aggregate

```
COUNT(*), SUM(x), AVG(x), MIN(x), MAX(x)
```

Window

```
ROW_NUMBER() OVER (PARTITION BY a ORDER BY b),
RANK(), DENSE_RANK(), LAG(x,1), LEAD(x,1),
NTILE(4)
```

JSON / XML / TRIGGERS

JSON / XML

```
JSON_VALUE(doc,'$.a'), JSON_TABLE(...)
XMLELEMENT(NAME a, x), XMLQUERY(...)
```

Trigger / Procedure

```
CREATE TRIGGER trg AFTER INSERT ON t
  FOR EACH ROW ...;
CREATE PROCEDURE p(IN x INT) ...;
```