SQL

Remy Wang

March 2026

Basic SQL

General Form

```
SELECT e1(x, y), e2(x, y)
  FROM t
 WHERE cond(x, y)
```

Meaning

```
for (x, y) in t:
  if cond(x, y):
    print(e1(x, y), e2(x, y))
```

Aggregate & Grouping

```
4:   SELECT x, y, AGG(z), AGG(w)
1:     FROM t
2:    WHERE cond(x, y, z, w)
3: GROUP BY x, y
```

1. Where's the data `FROM`?
2. Check conditions
3. Group rows
4. Run aggregate & return

Relational Algebra

Name	Notation	Meaning
selection	$\sigma_p(t)$	filter by condition p
projection	$\pi_{e(x,y)}(t)$	map an expression e over x, y
aggregation	$\gamma_{x,F(y)}(t)$	group by x, aggregate over y using F

Joins

```
SELECT ...
  FROM r, s
 WHERE cond
```

```
for x1, x2, ... in r:
  for y1, y2, ... in s:
    if cond(x1, x2, ..., y1, y2, ...):
      return ...
```

Cartesian Product

```
SELECT *
  FROM t, f
```

```
for t_row in t:
  for f_row in f:
    return t_row ++ f_row
```

In relational algebra: $\sigma_p(t \times f)$, or $t \bowtie_p f$ (the **join**).

Key Join

Lookup pet age

```
SELECT name, age
  FROM pet_name, pet_age
 WHERE pet_name.id = pet_age.id
```

General Form So Far

```
4.   SELECT R.x, AVG(T.z)
1.     FROM R, S, T
2.    WHERE R.x = S.x AND S.y = T.y
3. GROUP BY R.x
```

$$\gamma_{x,\text{avg}(T.z)}(\sigma_{R.x=S.x \wedge S.y=T.y}(R \times S \times T))$$

$$\gamma_{x,\text{avg}(T.z)}(R \bowtie_{R.x=S.x} S \bowtie_{S.y=T.y} T)$$

Subqueries

Find people with both cats & dogs

Wrong:

```
SELECT pets.person FROM pets
WHERE pets.kind="cat" AND pets.kind="dog";
```

```
SELECT pets.person FROM pets
WHERE pets.kind="cat" OR pets.kind="dog";
```

```
SELECT pets.person FROM pets
WHERE pets.kind="cat"
INTERSECT/UNION/EXCEPT
SELECT pets.person FROM pets
WHERE pets.kind="dog";
```

“Variables” in SQL

SQL	Python
`WITH`	Local variable
`CREATE TABLE`	Global variable
`CREATE VIEW`	Helper function
`CREATE MATERIALIZED VIEW`	Cached helper function

```
CREATE TABLE/(MATERIALIZED) VIEW cat_people AS
SELECT pets.person FROM pets
WHERE pets.kind="cat";
CREATE TABLE/(MATERIALIZED) VIEW dog_people AS
SELECT pets.person FROM pets
WHERE pets.kind="dog";

-- What happens if we run the following query twice?
-- What if we insert into `pets` then run the query?

-- INSERT INTO pets VALUES (...);

SELECT cat_people.person FROM cat_people
INTERSECT
SELECT dog_people.person FROM dog_people;
```

WITH has a slightly different syntax as it's local to the query:

```
WITH cat_people AS (
  SELECT pets.person FROM pets
  WHERE pets.kind="cat"
),   dog_people AS (
  SELECT pets.person FROM pets
  WHERE pets.kind="dog"
)
SELECT cat_people.person FROM cat_people
INTERSECT
SELECT dog_people.person FROM dog_people;
```

Find Pet Kinds w/ Avg. Age > 10?

Wrong:

```
SELECT kind, AVG(age) FROM pets
WHERE AVG(age) > 10
GROUP BY kind;
```

Two queries:

```sql
CREATE TABLE averages AS
SELECT kind, AVG(age) AS a
FROM pets
GROUP BY kind;

-- Now we just filter for over 10:
SELECT * FROM averages
WHERE averages.a > 10.0;
```

"One" query:

```
WITH averages AS (SELECT kind, AVG(age) AS a
                  FROM pets GROUP BY kind)
SELECT * FROM averages WHERE averages.a > 10.0;
```

ONE query:

```
SELECT kind, AVG(age) FROM pets
GROUP BY kind
HAVING AVG(age) > 10 -- condition on each group
```