

# Subqueries inside WHERE and SELECT clauses

JOINING DATA IN SQL

SQL

**Chester Ismay**

Data Science Evangelist, DataRobot

# Subquery inside WHERE clause set-up

name	indep_year	fert_rate	women_parli_perc
Australia	1901	1.88	32.74
Brunei	1984	1.96	6.06
Chile	1810	1.8	15.82
Egypt	1922	2.7	14.9
Haiti	1804	3.03	2.74
India	1947	2.43	11.58
Liberia	1847	4.64	11.65
Norway	1905	1.93	39.6
Oman	1951	2.75	8.82
Portugal	1143	1.31	34.8
Spain	1492	1.53	38.64
Uruguay	1828	2.03	22.31
Vietnam	1945	1.7	24

# Average fert\_rate

```
SELECT AVG(fert_rate)
FROM states;
```

```
+-----+
|      avg      |
+-----+
| 2.28385       |
+-----+
```

# Asian countries below average `fert\_rate`

```
SELECT name, fert_rate  
FROM states  
WHERE continent = 'Asia'
```

# Asian countries below average `fert\_rate`

```
SELECT name, fert_rate  
FROM states  
WHERE continent = 'Asia'  
      AND fert_rate <
```

# Asian countries below average `fert\_rate`

```
SELECT name, fert_rate
FROM states
WHERE continent = 'Asia'
      AND fert_rate <
      (SELECT AVG(fert_rate)
       FROM states);
```

# Asian countries below average `fert\_rate`

```
SELECT name, fert_rate
FROM states
WHERE continent = 'Asia'
      AND fert_rate <
      (SELECT AVG(fert_rate)
       FROM states);
```

```
+-----+-----+
| name   | fert_rate |
+-----+-----+
| Brunei | 1.96      |
| Vietnam | 1.7      |
+-----+-----+
```

# Subqueries inside SELECT clauses - setup

```
SELECT DISTINCT continent  
FROM prime_ministers;
```

```
+-----+  
| continent |  
+-----+  
| Africa    |  
| Asia      |  
| Europe    |  
| North America |  
| Oceania   |  
+-----+
```



# Subquery inside SELECT clause - complete

```
SELECT DISTINCT continent,  
  (SELECT COUNT(*)  
   FROM states  
   WHERE prime_ministers.continent = states.continent) AS countries_num  
FROM prime_ministers;
```

```
+-----+-----+  
| continent | countries_num |  
+-----+-----+  
| Africa    | 2             |  
| Asia      | 4             |  
| Europe    | 3             |  
| North America | 1           |  
| Oceania    | 1             |  
+-----+-----+
```

# Let's practice!

JOINING DATA IN SQL

# Subquery inside the FROM clause

JOINING DATA IN SQL

SQL

**Chester Ismay**

Data Science Evangelist, DataRobot

# Build-up

```
SELECT continent, MAX(women_parli_perc) AS max_perc
FROM states
GROUP BY continent
ORDER BY continent;
```

```
+-----+-----+
| continent | max_perc |
+-----+-----+
| Africa    | 14.9     |
| Asia      | 24       |
| Europe    | 39.6     |
| North America | 2.74    |
| Oceania   | 32.74    |
| South America | 22.31   |
+-----+-----+
```

# Focusing on records in monarchs

```
SELECT monarchs.continent
FROM monarchs, states
WHERE monarchs.continent = states.continent
ORDER BY continent;
```

```
+-----+
| continent |
+-----+
| Asia      |
| Asia      |
| Asia      |
| Asia      |
| Asia      |
| Asia      |
| Asia      |
| Asia      |
| Asia      |
| Asia      |
| Europe    |
| Europe    |
| Europe    |
| Europe    |
| Europe    |
| Europe    |
+-----+
```

# Finishing off the subquery

```
SELECT DISTINCT monarchs.continent, subquery.max_perc
FROM monarchs,
    (SELECT continent, MAX(women_parli_perc) AS max_perc
     FROM states
     GROUP BY continent) AS subquery
WHERE monarchs.continent = subquery.continent
ORDER BY continent;
```

```
+-----+-----+
| continent | max_perc |
+-----+-----+
| Asia      | 24       |
| Europe    | 39.6     |
+-----+-----+
```

# Let's practice!

JOINING DATA IN SQL

# Course Review

JOINING DATA IN SQL



**Chester Ismay**

Data Science Evangelist, DataRobot



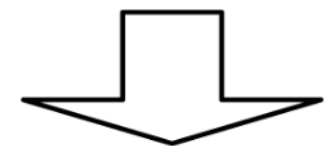
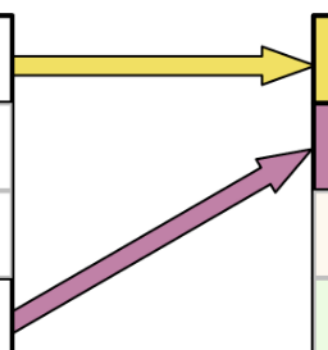
# Types of joins

- INNER JOIN
  - Self-joins
- OUTER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - FULL JOIN
- CROSS JOIN
- Semi-join / Anti-join

# INNER JOIN vs LEFT JOIN

left\_table      right\_table

id	val	id	val
1	L1	1	R1
2	L2	4	R2
3	L3	5	R3
4	L4	6	R4

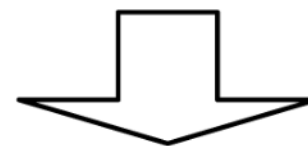
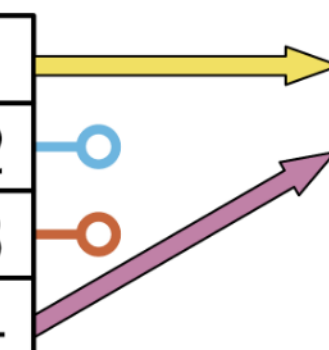


INNER JOIN

L_id	L_val	R_val
1	L1	R1
4	L4	R2

left\_table      right\_table

id	val	id	val
1	L1	1	R1
2	L2	4	R2
3	L3	5	R3
4	L4	6	R4



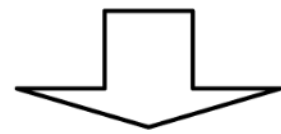
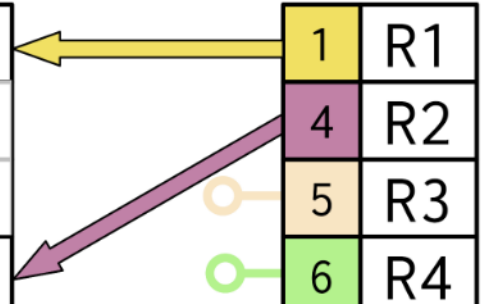
LEFT JOIN

L_id	L_val	R_val
1	L1	R1
2	L2	
3	L3	
4	L4	R2

# RIGHT JOIN vs FULL JOIN

left\_table      right\_table

id	val	id	val
1	L1	1	R1
2	L2	4	R2
3	L3	5	R3
4	L4	6	R4

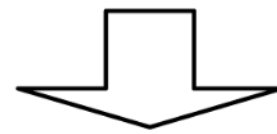
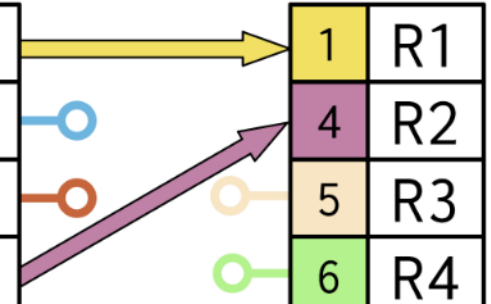


RIGHT JOIN

R_id	L_val	R_val
1	L1	R1
4	L4	R2
5		R3
6		R4

left\_table      right\_table

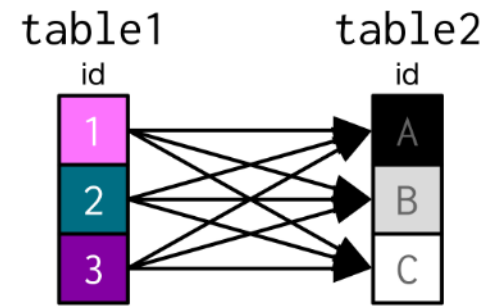
id	val	id	val
1	L1	1	R1
2	L2	4	R2
3	L3	5	R3
4	L4	6	R4



FULL JOIN

L_id	R_id	L_val	R_val
1	1	L1	R1
2		L2	
3		L3	
4	4	L4	R2
	5		R3
	6		R4

# CROSS JOIN with code

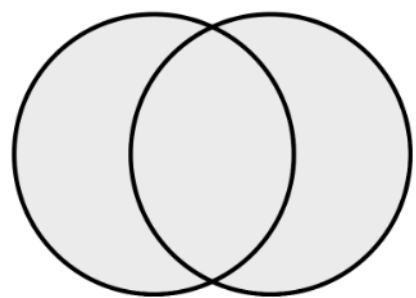


CROSS JOIN

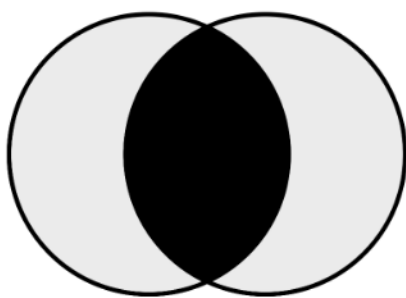
id1	id2
1	A
1	B
1	C
2	A
2	B
2	C
3	A
3	B
3	C

```
SELECT table1.id AS id1,  
       table2.id AS id2  
FROM table1  
CROSS JOIN table2;
```

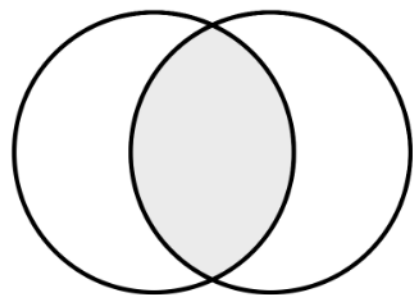
# Set Theory Clauses



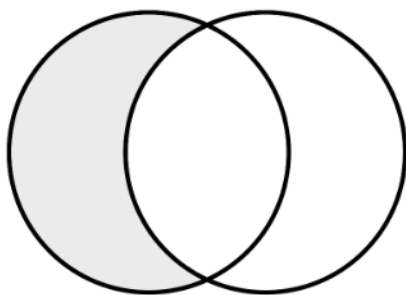
UNION



UNION ALL



INTERSECT



EXCEPT

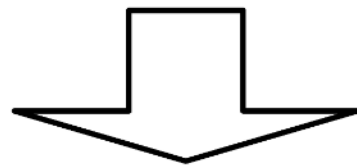
# Semi-joins and Anti-joins

left\_table

id	val
1	L1
2	L2
3	L3
4	L4

right\_table

id	val
1	R1
4	R2
5	R3
6	R4



Semi-join

L\_id L\_val

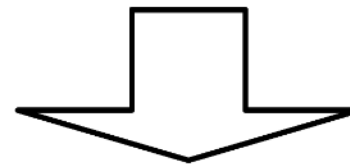
1	L1
4	L4

left\_table

id	val
1	L1
2	L2
3	L3
4	L4

right\_table

id	val
1	R1
4	R2
5	R3
6	R4



Anti-join

L\_id L\_val

2	L2
3	L3

# Types of basic subqueries

- Subqueries inside WHERE clauses
- Subqueries inside SELECT clauses
- Subqueries inside FROM clauses

**Own the challenge  
problems! You got  
this!**

**JOINING DATA IN SQL**



