Data Analysis 3: Window Functions

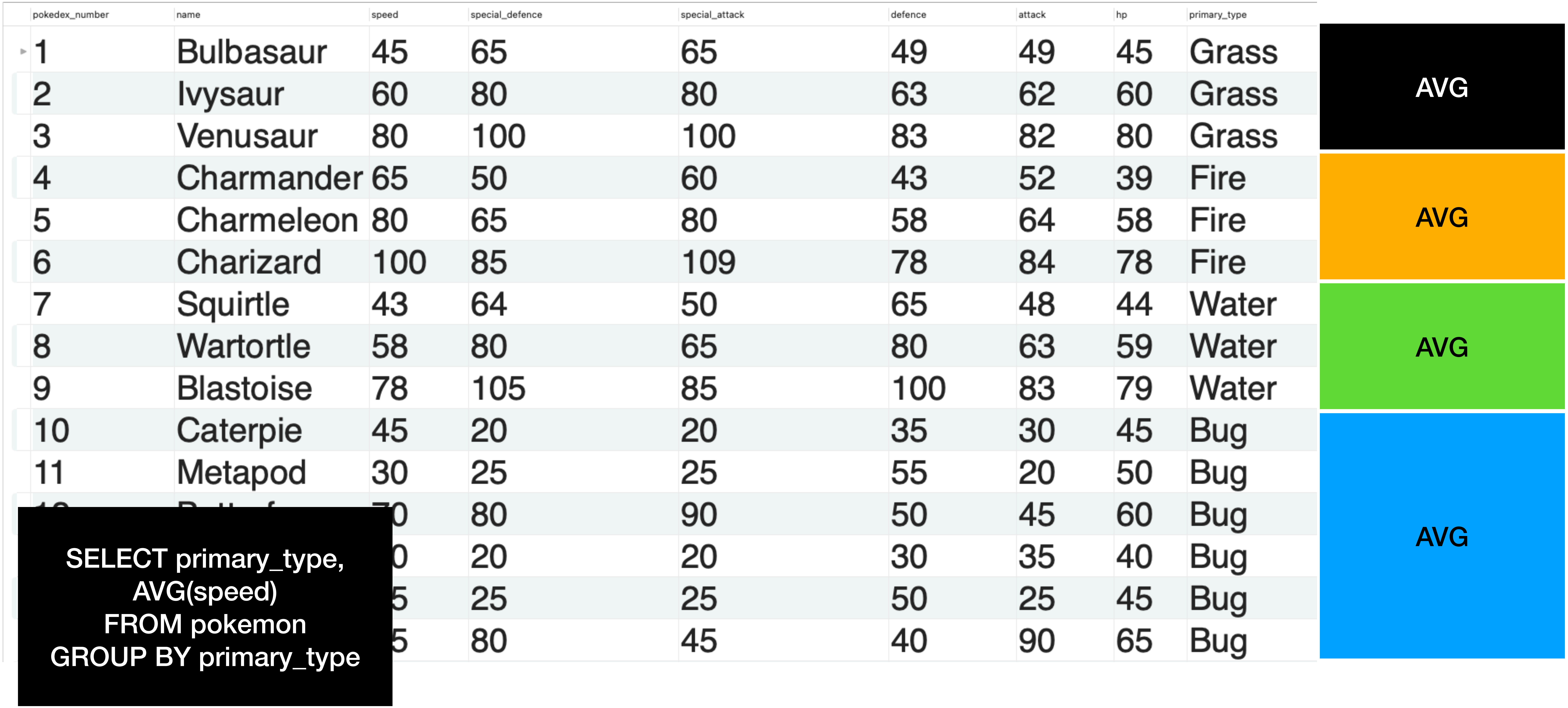
Dataforståelse

NIFR 2024

Data analysis 4

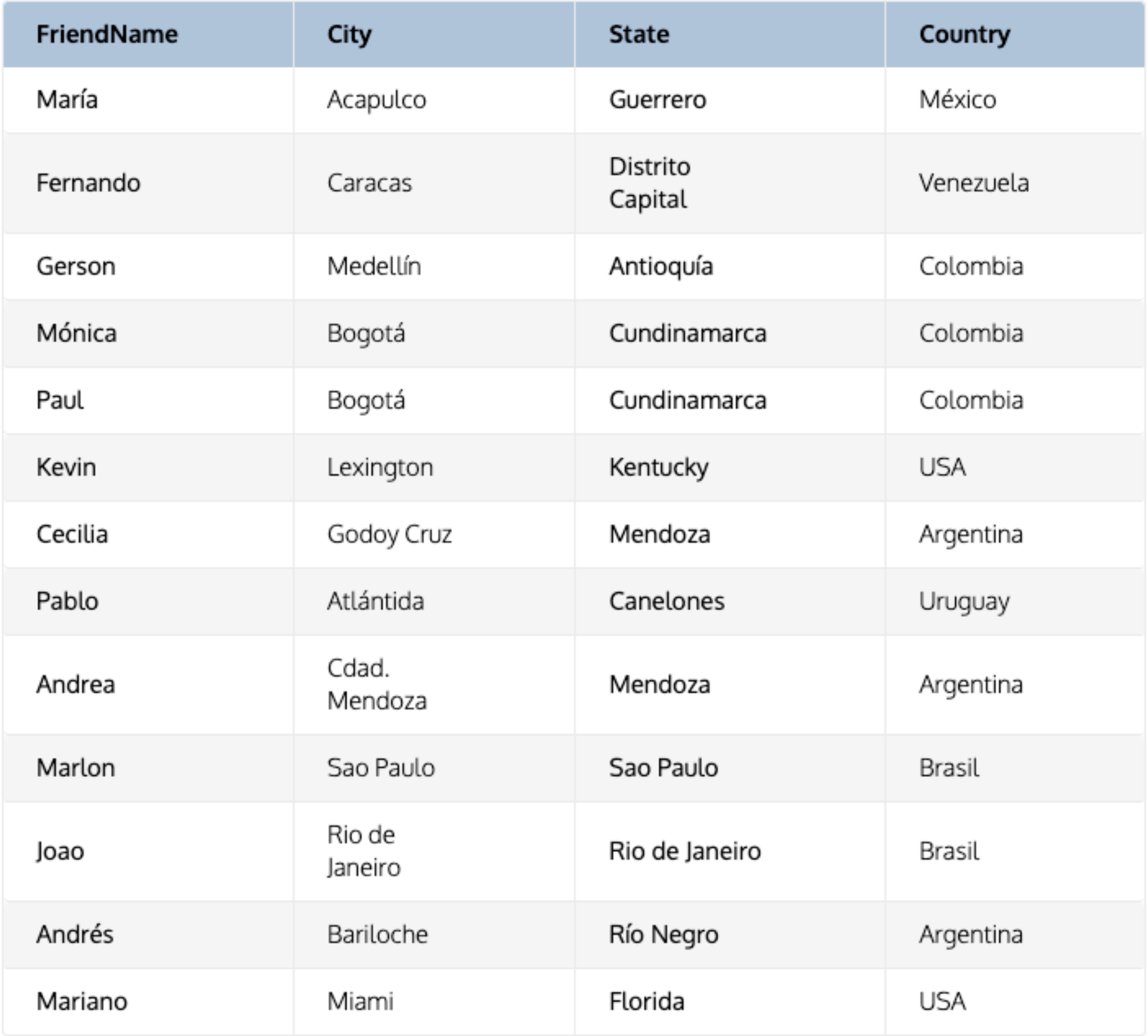
Dataforståelse

- Aggregate functions
  - Combining aggregate and ‘regular’ values
- The ‘OVER’ keyword
- The ‘WITH’ keyword



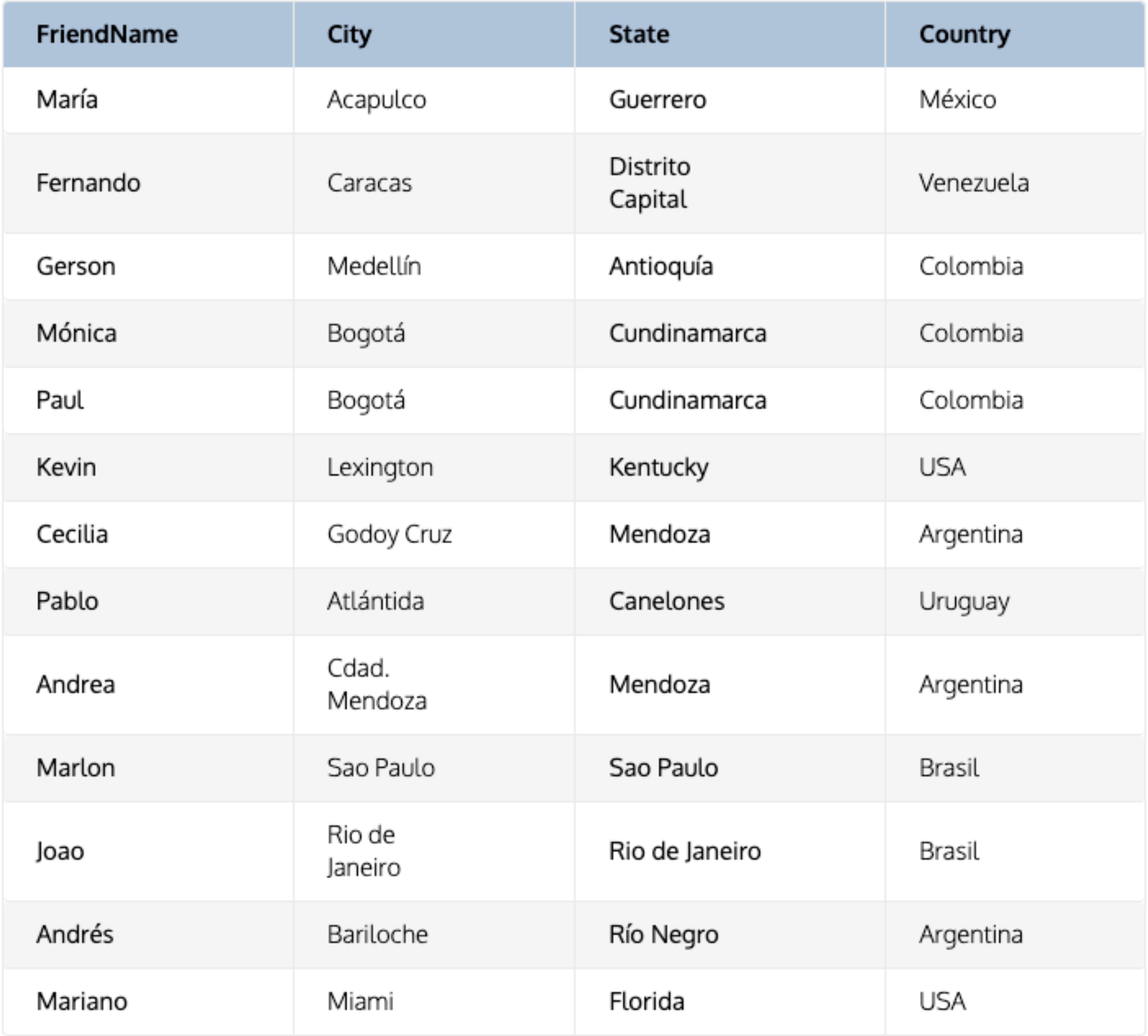
pokedex_number	name	speed	special_defence	special_attack	defence	attack	hp	primary_type
1	Bulbasaur	45	65	65	49	49	45	Grass
2	Ivysaur	60	80	80	63	62	60	Grass
3	Venusaur	80	100	100	83	82	80	Grass
4	Charmander	65	50	60	43	52	39	Fire
5	Charmeleon	80	65	80	58	64	58	Fire
6	Charizard	100	85	109	78	84	78	Fire
7	Squirtle	43	64	50	65	48	44	Water
8	Wartortle	58	80	65	80	63	59	Water
9	Blastoise	78	105	85	100	83	79	Water
10	Caterpie	45	20	20	35	30	45	Bug
11	Metapod	30	25	25	55	20	50	Bug
12	[illegible]	70	80	90	50	45	60	Bug
[illegible]	[illegible]	[illegible]	20	20	30	35	40	Bug
[illegible]	[illegible]	[illegible]	25	25	50	25	45	Bug
[illegible]	[illegible]	[illegible]	80	45	40	90	65	Bug

```
SELECT primary_type,
AVG(speed)
FROM pokemon
GROUP BY primary_type
```

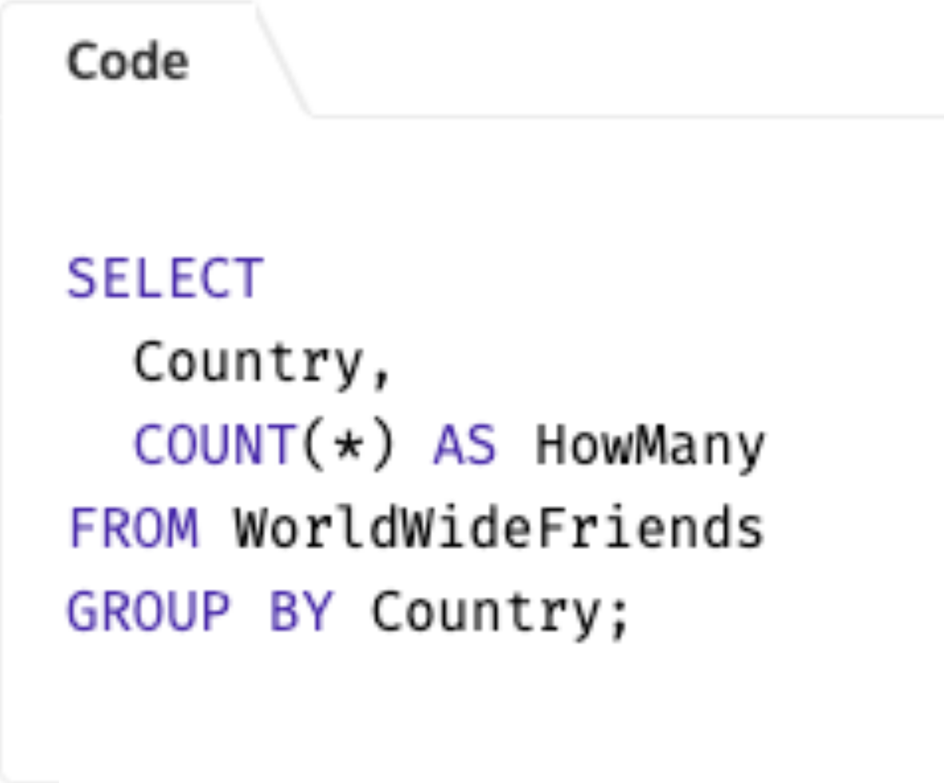


FriendName	City	State	Country
María	Acapulco	Guerrero	México
Fernando	Caracas	Distrito Capital	Venezuela
Gerson	Medellín	Antioquía	Colombia
Mónica	Bogotá	Cundinamarca	Colombia
Paul	Bogotá	Cundinamarca	Colombia
Kevin	Lexington	Kentucky	USA
Cecilia	Godoy Cruz	Mendoza	Argentina
Pablo	Atlántida	Canelones	Uruguay
Andrea	Cdad. Mendoza	Mendoza	Argentina
Marlon	Sao Paulo	Sao Paulo	Brasil
Joao	Rio de Janeiro	Rio de Janeiro	Brasil
Andrés	Bariloche	Río Negro	Argentina
Mariano	Miami	Florida	USA

https://learnsql.com/blog/group-by-multiple-columns/

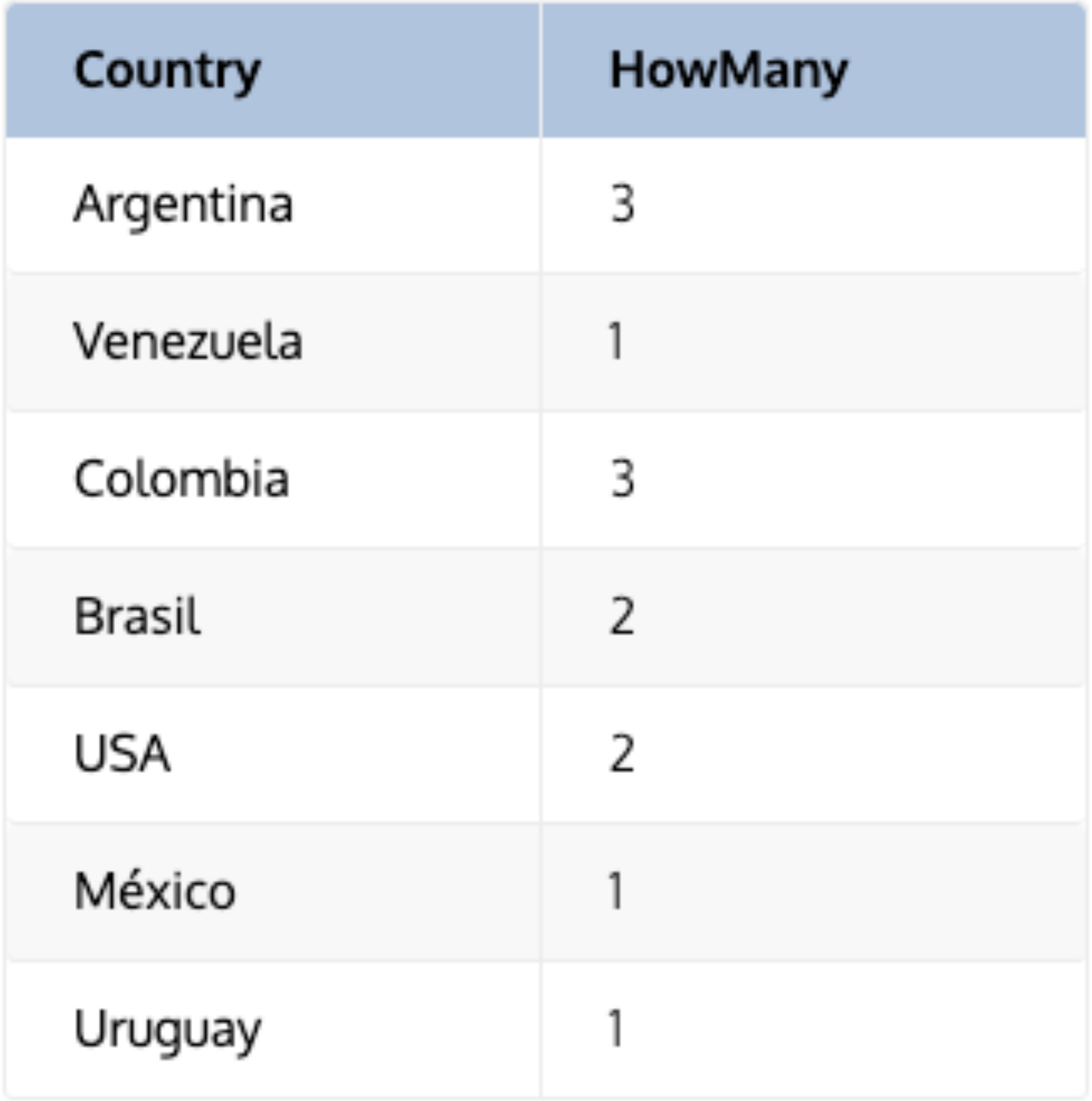


FriendName	City	State	Country
María	Acapulco	Guerrero	México
Fernando	Caracas	Distrito Capital	Venezuela
Gerson	Medellín	Antioquía	Colombia
Mónica	Bogotá	Cundinamarca	Colombia
Paul	Bogotá	Cundinamarca	Colombia
Kevin	Lexington	Kentucky	USA
Cecilia	Godoy Cruz	Mendoza	Argentina
Pablo	Atlántida	Canelones	Uruguay
Andrea	Cdad. Mendoza	Mendoza	Argentina
Marlon	Sao Paulo	Sao Paulo	Brasil
Joao	Rio de Janeiro	Rio de Janeiro	Brasil
Andrés	Bariloche	Río Negro	Argentina
Mariano	Miami	Florida	USA



```
SELECT
  Country,
  COUNT(*) AS HowMany
FROM WorldWideFriends
GROUP BY Country;
```

Aggregated Columns

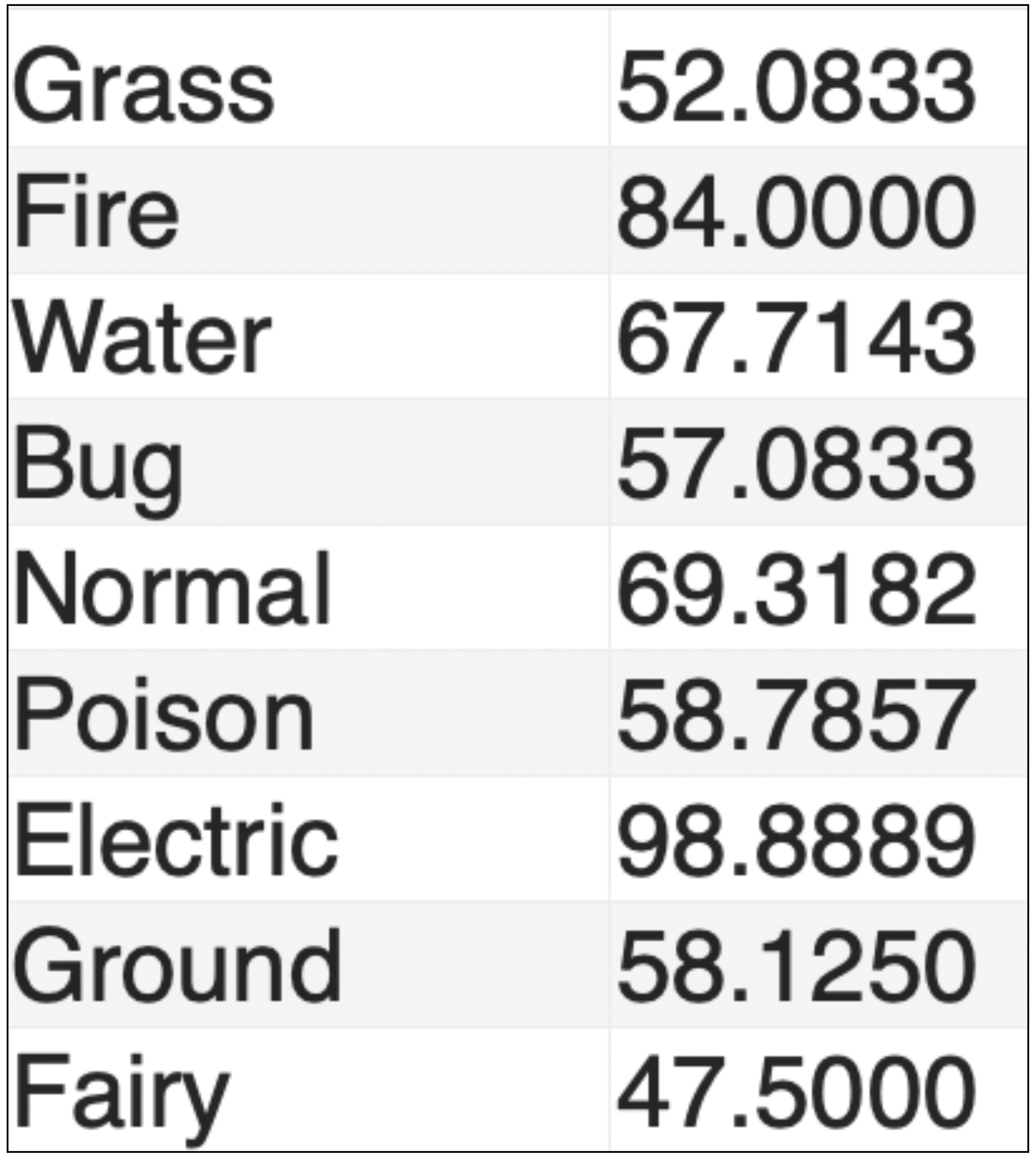


Country	HowMany
Argentina	3
Venezuela	1
Colombia	3
Brasil	2
USA	2
México	1
Uruguay	1

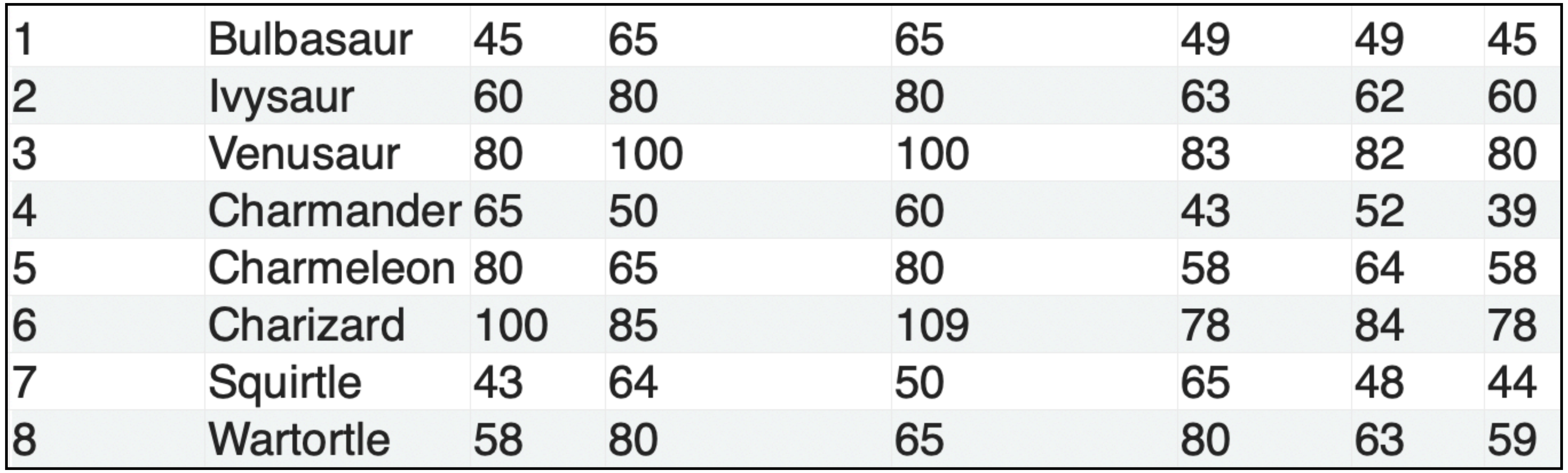
Result

https://learnsql.com/blog/group-by-multiple-columns/

Aggregated Columns



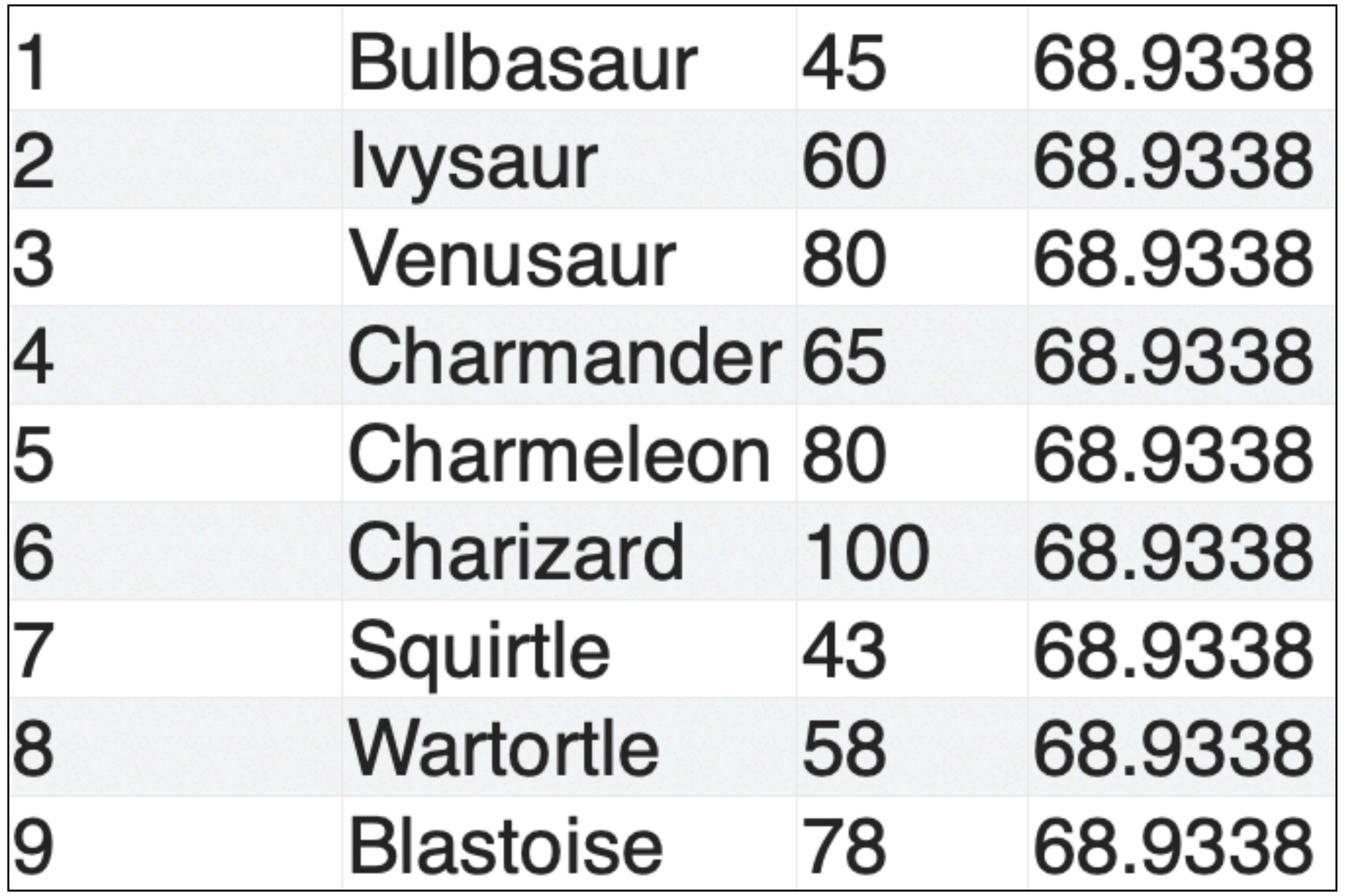
Grass	52.0833
Fire	84.0000
Water	67.7143
Bug	57.0833
Normal	69.3182
Poison	58.7857
Electric	98.8889
Ground	58.1250
Fairy	47.5000



1	Bulbasaur	45	65	65	49	49	45
2	Ivysaur	60	80	80	63	62	60
3	Venusaur	80	100	100	83	82	80
4	Charmander	65	50	60	43	52	39
5	Charmeleon	80	65	80	58	64	58
6	Charizard	100	85	109	78	84	78
7	Squirtle	43	64	50	65	48	44
8	Wartortle	58	80	65	80	63	59

Non-aggregated

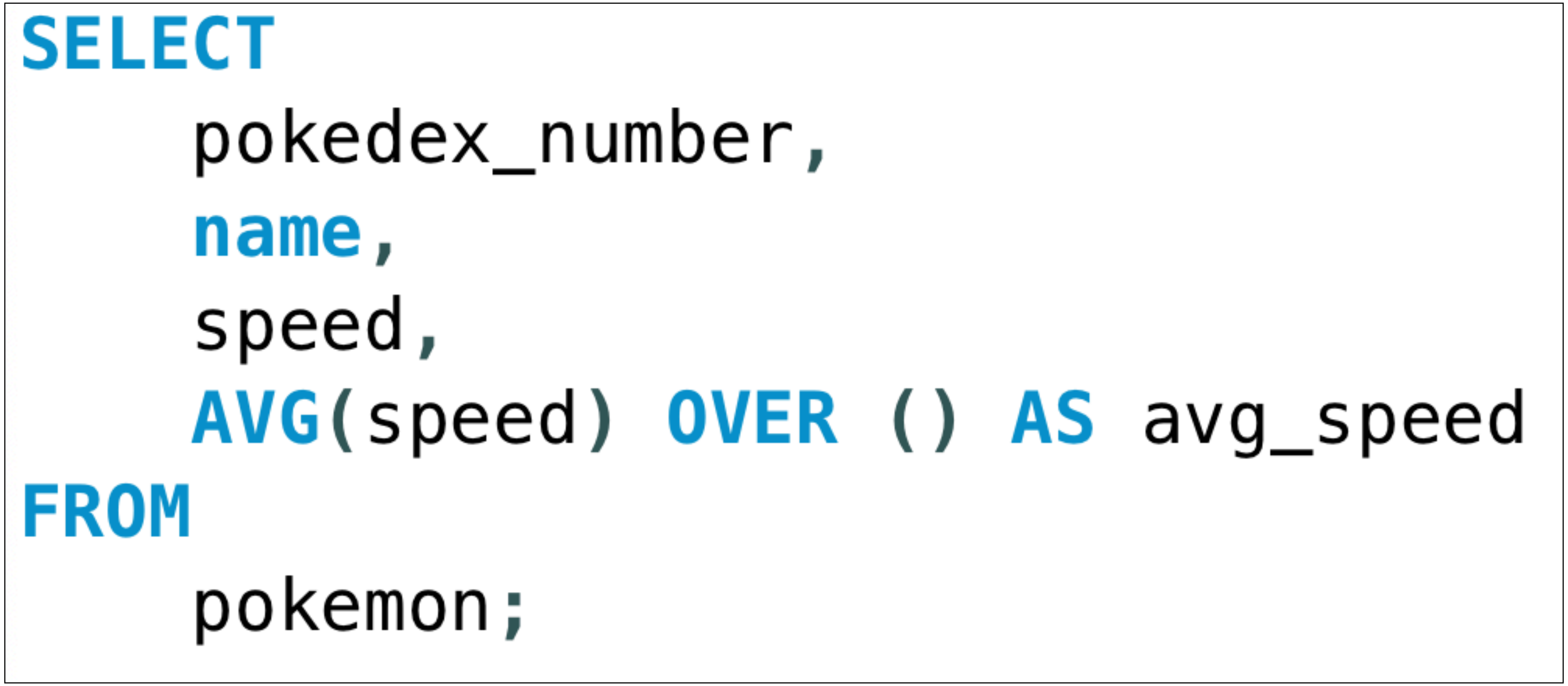
Issue: Comparing a single entities value with the combined average



1	Bulbasaur	45	68.9338
2	Ivysaur	60	68.9338
3	Venusaur	80	68.9338
4	Charmander	65	68.9338
5	Charmeleon	80	68.9338
6	Charizard	100	68.9338
7	Squirtle	43	68.9338
8	Wartortle	58	68.9338
9	Blastoise	78	68.9338

Window function

Combining aggregated columns & non-aggregated columns

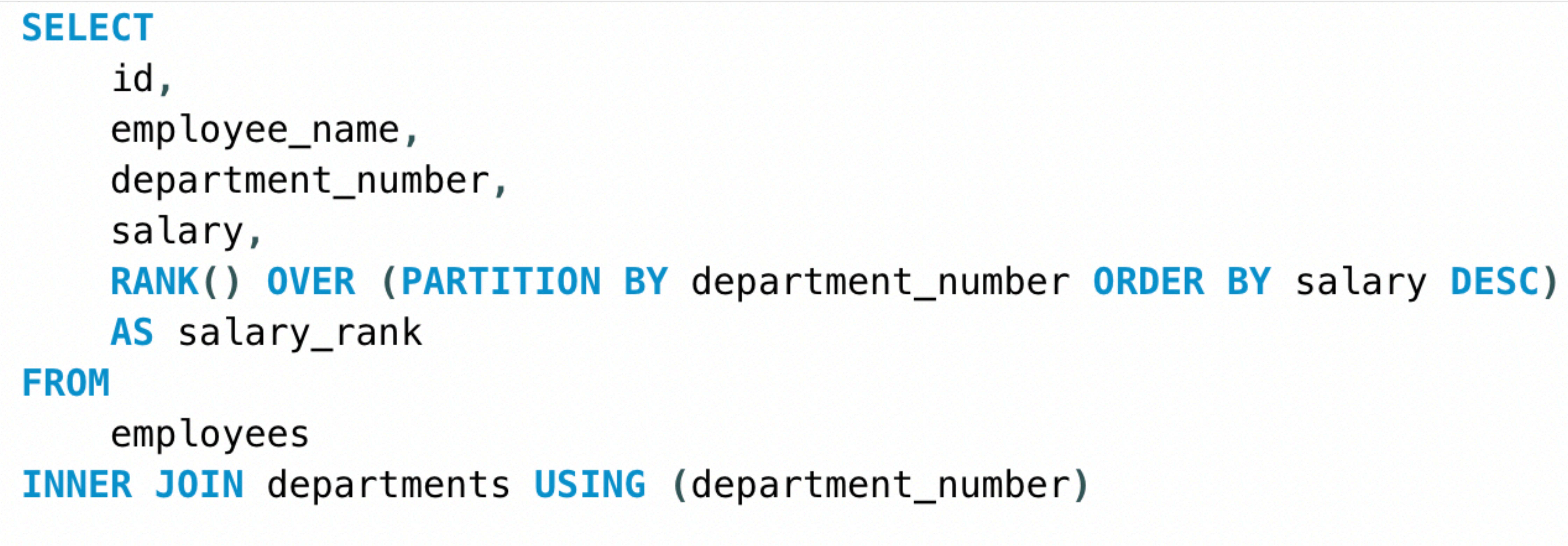


```
SELECT
    pokedex_number,
    name,
    speed,
    AVG(speed) OVER () AS avg_speed
FROM
    pokemon;
```

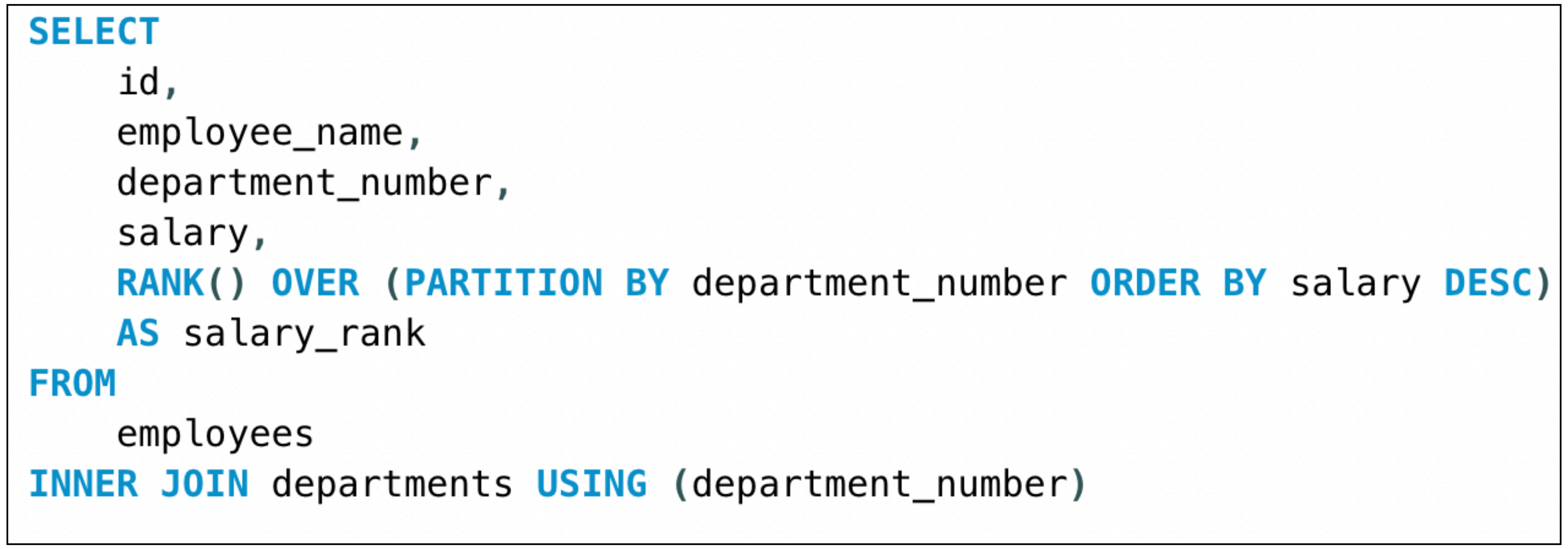
Example: Partitions & Partion order

Window function

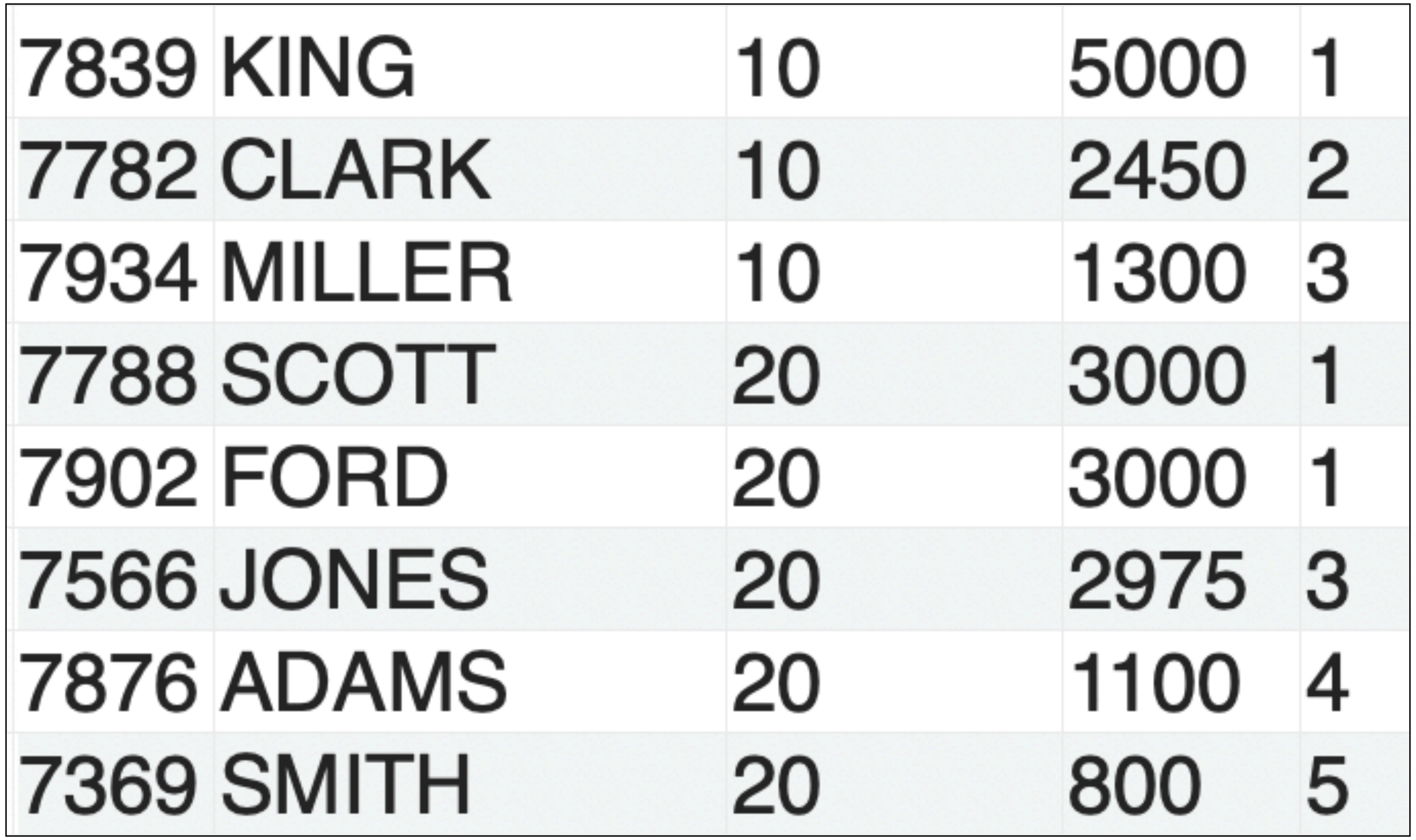
Ranking results



```
SELECT
    id,
    employee_name,
    department_number,
    salary,
    RANK() OVER (PARTITION BY department_number ORDER BY salary DESC)
    AS salary_rank
FROM
    employees
INNER JOIN departments USING (department_number)
```



```
SELECT
    id,
    employee_name,
    department_number,
    salary,
    RANK() OVER (PARTITION BY department_number ORDER BY salary DESC)
    AS salary_rank
FROM
    employees
INNER JOIN departments USING (department_number)
```



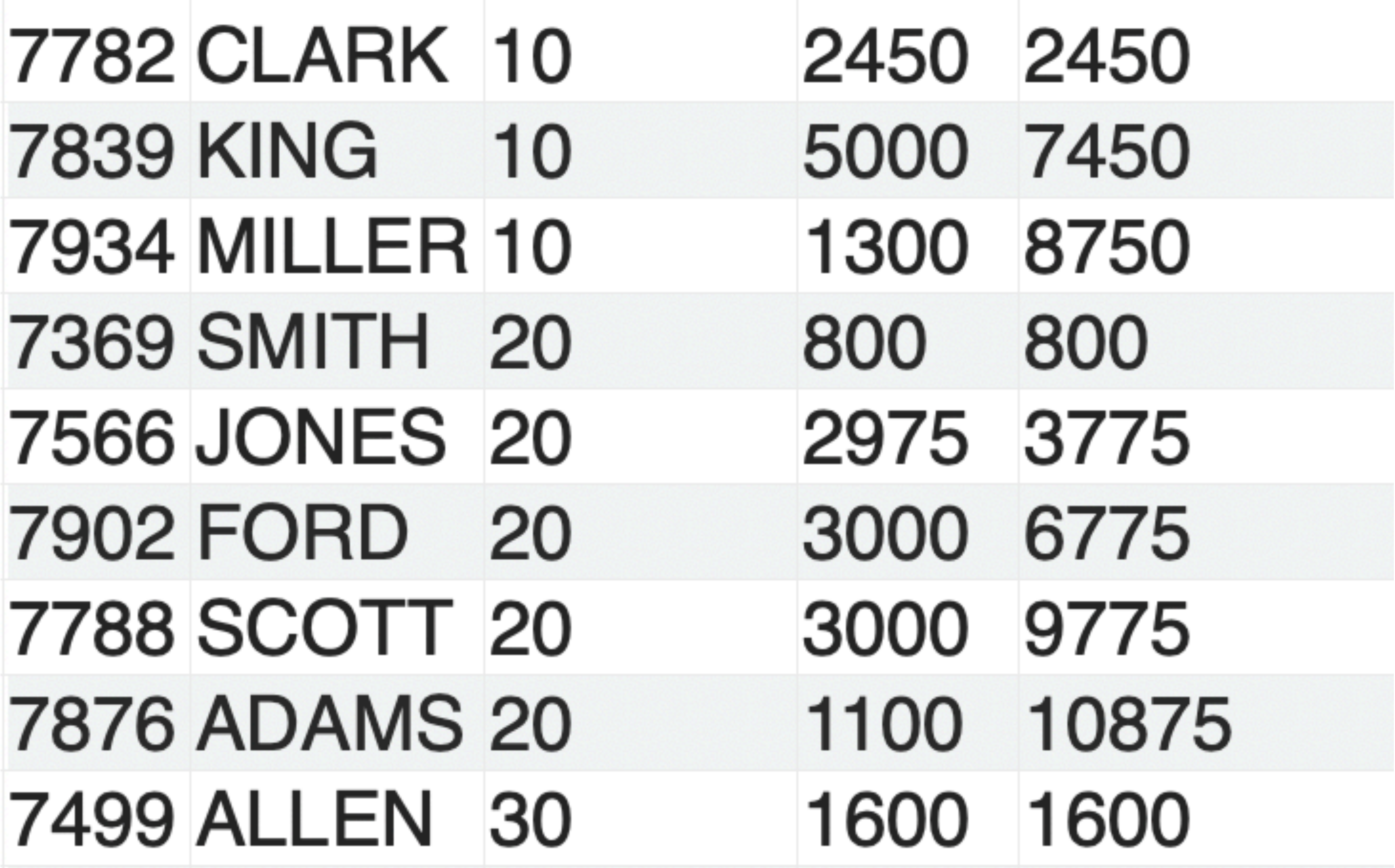
7839	KING	10	5000	1
7782	CLARK	10	2450	2
7934	MILLER	10	1300	3
7788	SCOTT	20	3000	1
7902	FORD	20	3000	1
7566	JONES	20	2975	3
7876	ADAMS	20	1100	4
7369	SMITH	20	800	5

Exercises 1 A & B

Example: Removing rows before partitioning

Window function

Running total

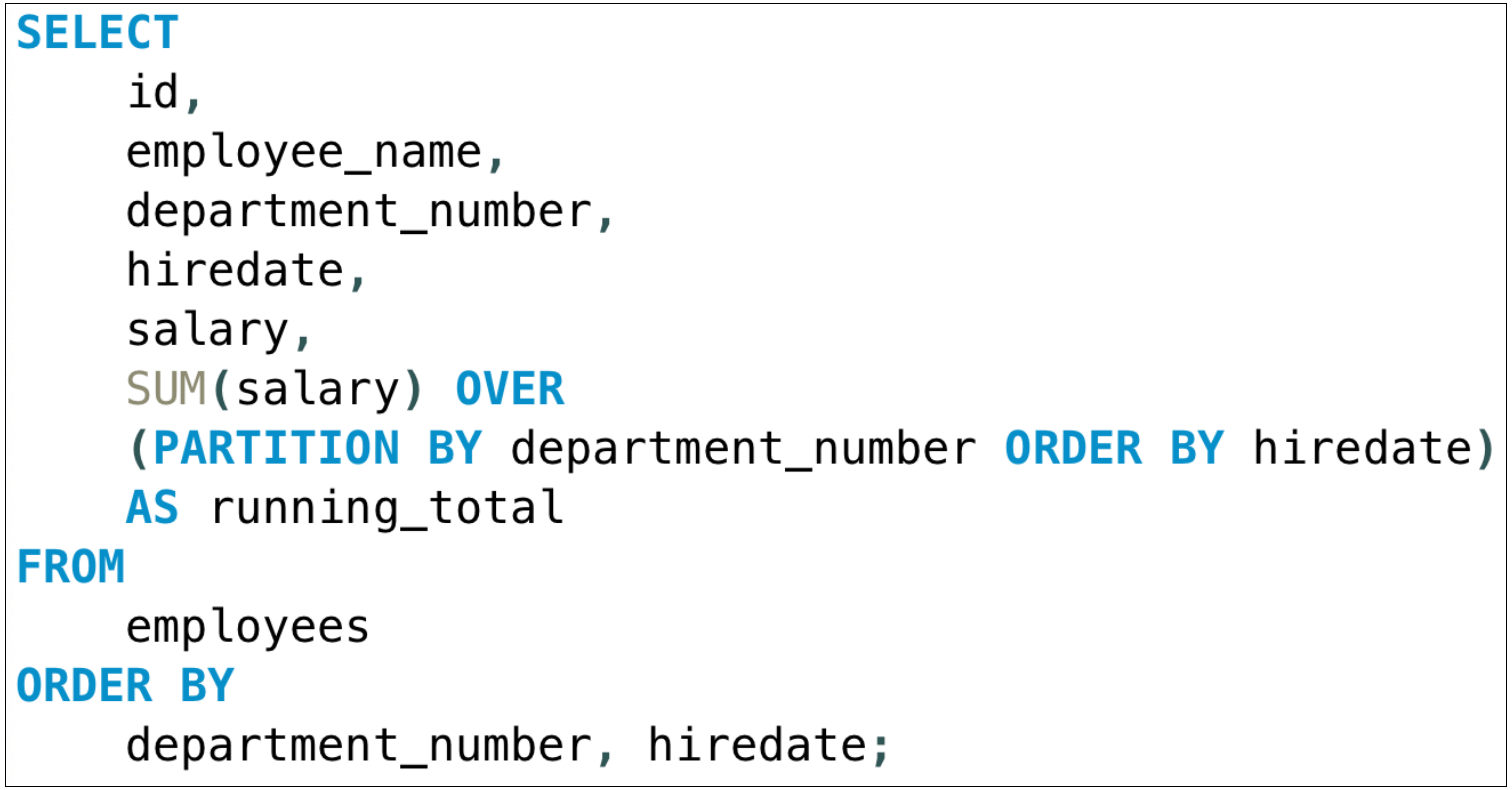


7782	CLARK	10	2450	2450
7839	KING	10	5000	7450
7934	MILLER	10	1300	8750
7369	SMITH	20	800	800
7566	JONES	20	2975	3775
7902	FORD	20	3000	6775
7788	SCOTT	20	3000	9775
7876	ADAMS	20	1100	10875
7499	ALLEN	30	1600	1600

Running total

Window function

Running total

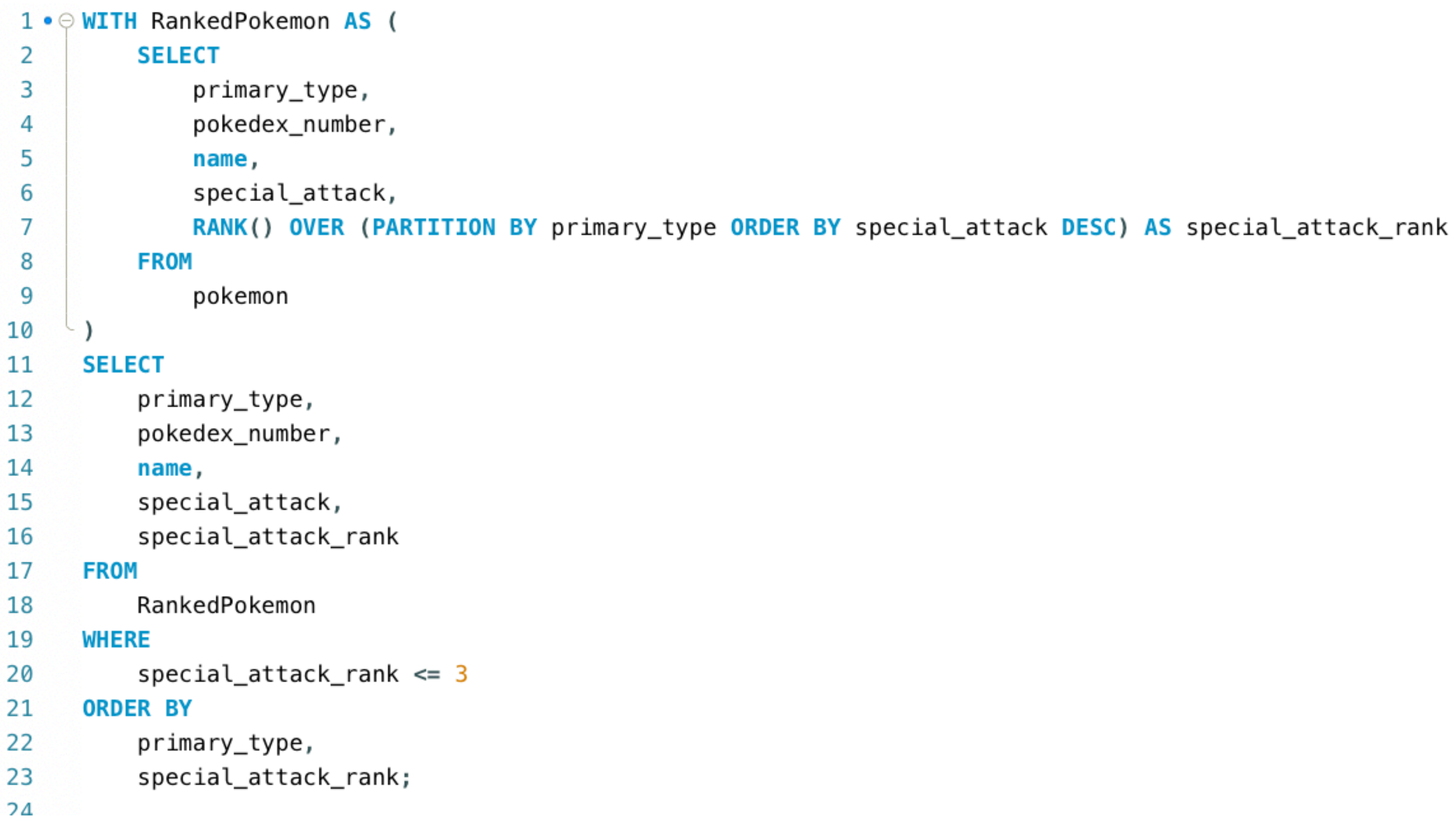


```sql
SELECT
    id,
    employee_name,
    department_number,
    hiredate,
    salary,
    SUM(salary) OVER
    (PARTITION BY department_number ORDER BY hiredate)
    AS running_total
FROM
    employees
ORDER BY
    department_number, hiredate;
```

Running total

SubQueries and Window Functions

‘With’ Keyword

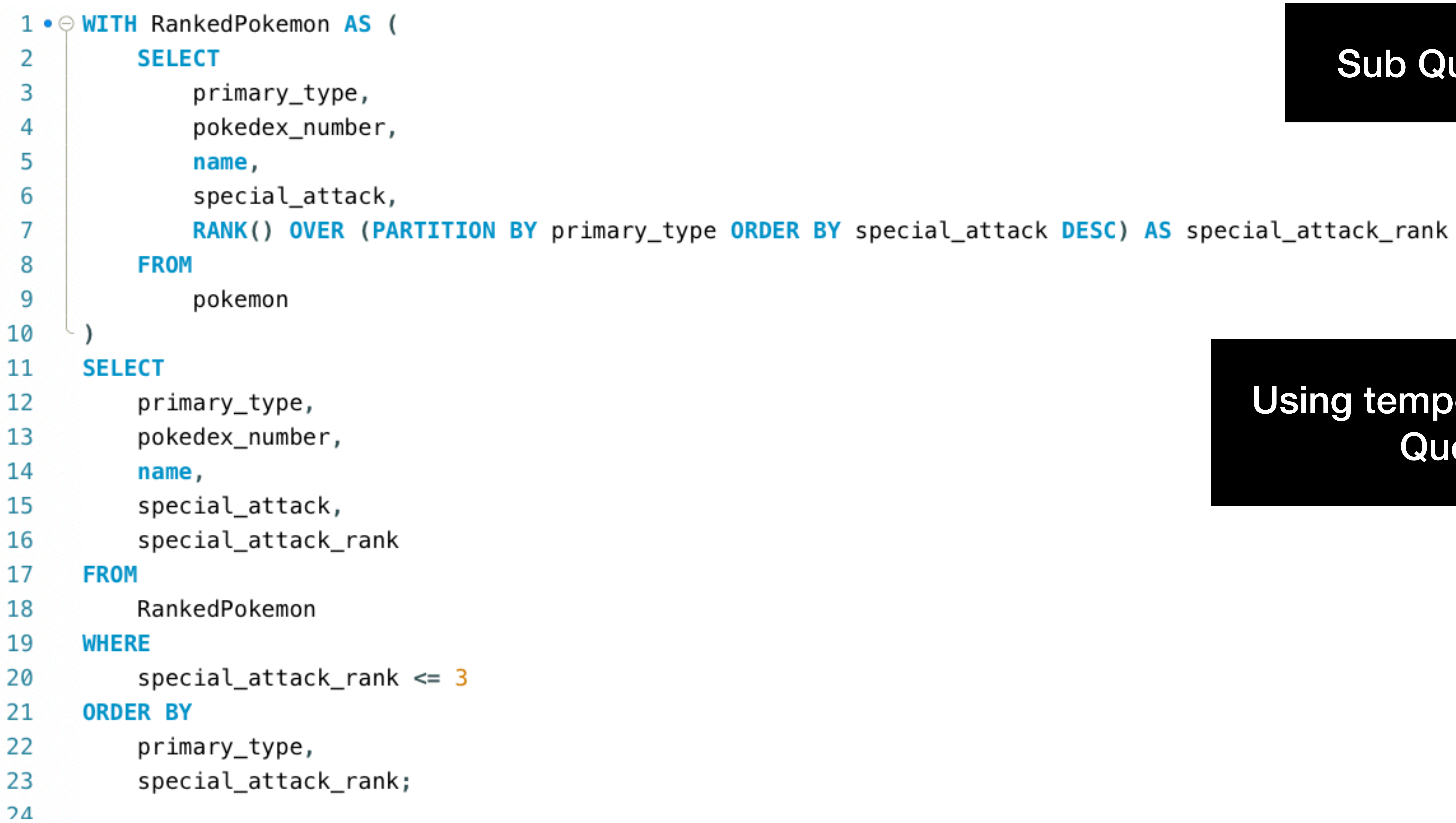


```sql
WITH RankedPokemon AS (
    SELECT
        primary_type,
        pokedex_number,
        name,
        special_attack,
        RANK() OVER (PARTITION BY primary_type ORDER BY special_attack DESC) AS special_attack_rank
    FROM
        pokemon
)
SELECT
    primary_type,
    pokedex_number,
    name,
    special_attack,
    special_attack_rank
FROM
    RankedPokemon
WHERE
    special_attack_rank <= 3
ORDER BY
    primary_type,
    special_attack_rank;
```

New Column

SubQueries and Window Functions

‘With’ Keyword



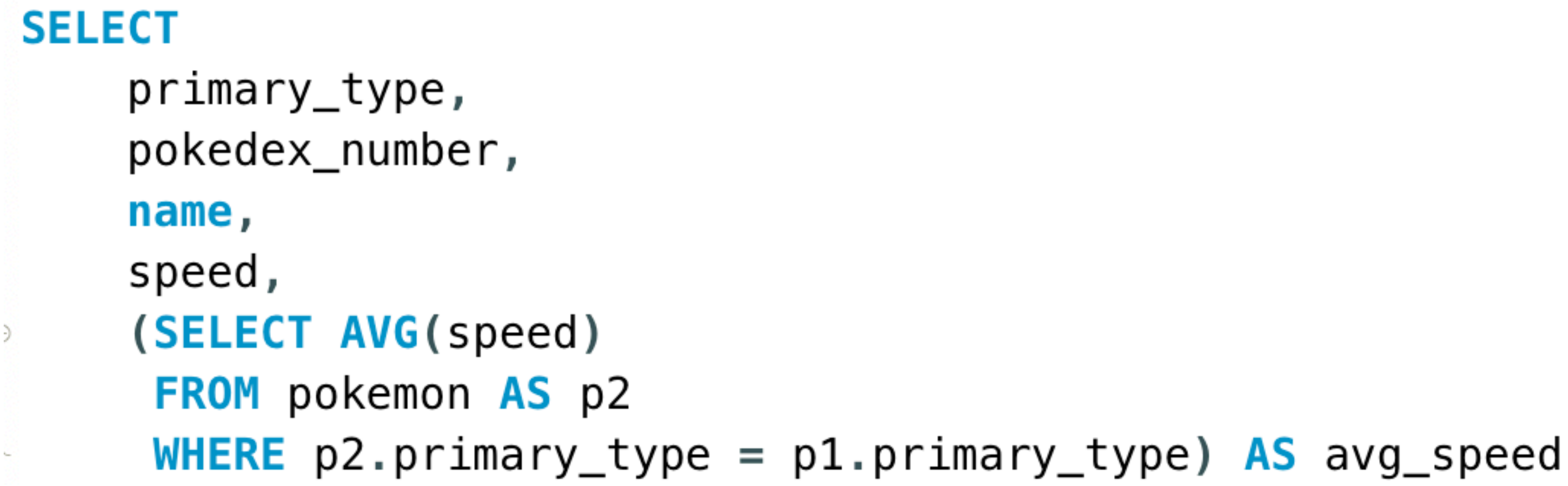
```sql
WITH RankedPokemon AS (
    SELECT
        primary_type,
        pokedex_number,
        name,
        special_attack,
        RANK() OVER (PARTITION BY primary_type ORDER BY special_attack DESC) AS special_attack_rank
    FROM
        pokemon
)
SELECT
    primary_type,
    pokedex_number,
    name,
    special_attack,
    special_attack_rank
FROM
    RankedPokemon
WHERE
    special_attack_rank <= 3
ORDER BY
    primary_type,
    special_attack_rank;
```

Sub Query

Using temporary result Query

SubQueries as columns

Aggregate functions



```sql
SELECT
    primary_type,
    pokedex_number,
    name,
    speed,
    (SELECT AVG(speed)
     FROM pokemon AS p2
     WHERE p2.primary_type = p1.primary_type) AS avg_speed
```

Exercises 1: C & D

Exercises 2

Sakila Dataset