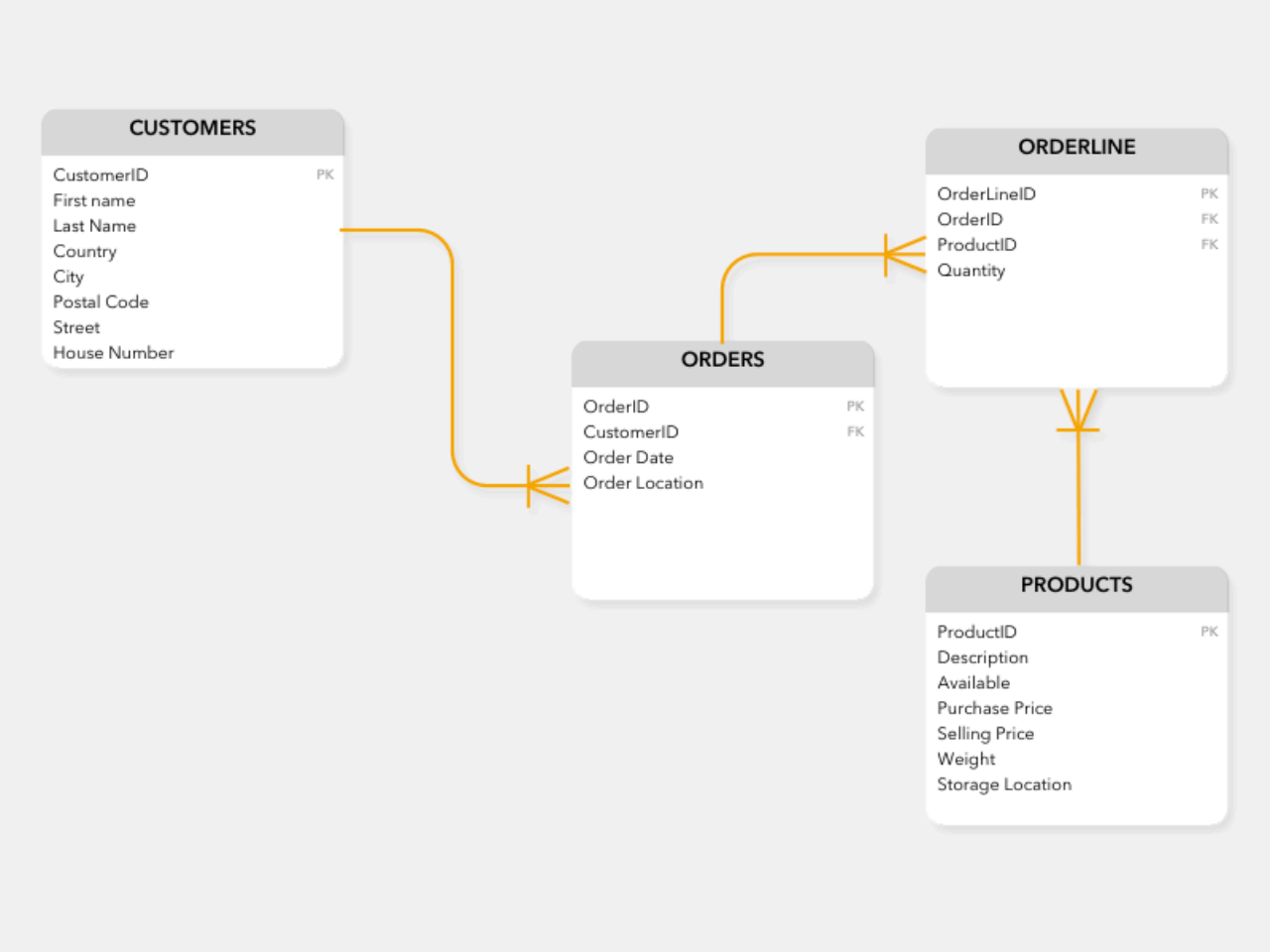
Advanced SQL

⏪ Recap

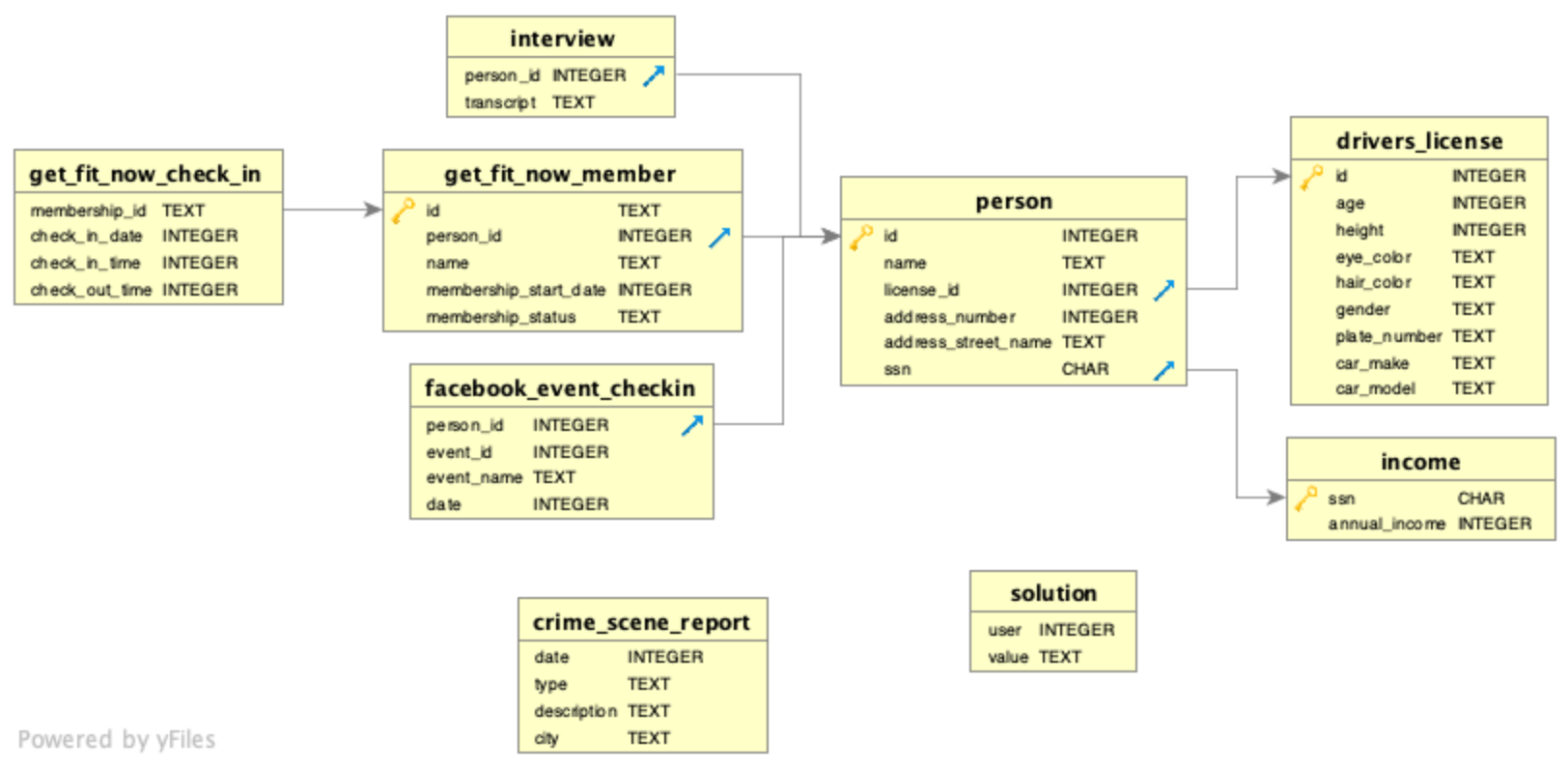
- `SELECT` columns `FROM` a table `WHERE` conditions are true
  - `LIKE` pattern matching operator (with `%` and `_`)
  - `IN` membership operator
  - `BETWEEN` range operator
- `LIMIT` the number of records returned
- `ORDER BY` columns

How Relationships Work in ERD

- Relationships are defined using **keys**
- **Primary Key (PK)**
  Unique ID for each record in an entity
- **Foreign Key (FK)**
  A field in one entity that links to the primary key of another entity



⏪ Recap



`get_fit_now_member` ➡️ `drivers_license`?

`facebook_event_checkin` ➡️ `income`?

⏪ Recap

```
select * from person
-- suspect person_id from first witness
where id = 67318
-- suspect person_id from second witness
and id in (67318, 28819)
```

The Suspect

id	person_id	name	membership_start_date	membership_status
48Z55	67318	Jeremy Bowers	20160101	gold

```
select * from interview where person_id=67318
```

> I was hired by a woman with a lot of money. I don't know her name but I know she's around 5'5" (65") or 5'7" (67"). She has red hair and she drives a Tesla Model S. I know that she attended the SQL Symphony Concert 3 times in December 2017.

Clue #1: Find the person who matches the description

> I don't know her name but I know she's around 5'5" (65") or 5'7" (67"). She has red hair and she drives a Tesla Model S.

Clue #1: Find the person who matches the description

```
select *
from drivers_license
where hair_color = 'red'
and car_make = 'Tesla'
and car_model = 'Model S'
and height between 65 and 67
```

Cross-reference with `person` table

id	age	height	eye_color	hair_color	gender	plate_number	car_make
202298	68	66	green	red	female	500123	Tesla
291182	65	66	blue	red	female	08CM64	Tesla
918773	48	65	black	red	female	917UU3	Tesla

`WHERE` column `IN` (list of values)

```
select *
from person
where license_id in (918773, 291182, 202298)
```

Cross-reference using Subquery

Subquery:

- A query within a query
- Returns the result of the inner query to the outer query
- Similar to how functions return values in Python

Cross-reference using Subquery

1️⃣ Query to get the list of IDs

```
select id
from drivers_license
where hair_color = 'red'
and car_make = 'Tesla'
and car_model = 'Model S'
and height between 65 and 67
```

id
202298
291182
918773

Cross-reference using Subquery

2️⃣ Use the query in the `WHERE` clause (i.e., subquery)

`WHERE` column `IN` (subquery returning a list of one column)

```
select *
from person
-- Instead of hard-coding the list of IDs
-- where license_id in (918773, 291182, 202298)
-- Use the subquery that returns that list
where license_id in (
    select id
    from drivers_license
    where hair_color = 'red'
    and car_make = 'Tesla'
    and car_model = 'Model S'
    and height between 65 and 67
)
```

Cross-reference using Subquery

Number of columns in subquery must match the number of columns in the WHERE clause

`WHERE` (column1, column2) `IN` (subquery returning a list of column1 & column2)

```
select *
from person
-- WHERE expects two columns: license_id and name
where (license_id, name) in (
    -- Subquery returns the list of the two columns that match: id and name
    select id, name
    from drivers_license
    where hair_color = 'red'
    and car_make = 'Tesla'
    and car_model = 'Model S'
    and height between 65 and 67
)
```

Use Subquery to Solve the Mystery

1. Find the interview of the first witness, who lives at the last house on "Northwestern Dr."

1. Find the interview of the first witness, who lives at the last house on "Northwestern Dr."

1️⃣ Query to get the person ID of the last house on Northwestern Dr.

```
select id from person
where address_street_name = 'Northwestern Dr'
order by address_number DESC limit 1
```

2️⃣ Use the query to get the interview

```
select * from interview
where person_id = (
    -- same as above
    select id from person
    where address_street_name = 'Northwestern Dr'
    order by address_number DESC limit 1
    )
```

🖥️ 2. Find the first suspect from person table: membership id starts with "48Z" and license plate includes "H42W"

🖥️ 3. Pull up the interview of the second witness. Her name is Annabel and she lives somewhere on "Franklin Ave"

7. From first witness: gym membership id starts with "48Z" and license plate includes "H42W"
8. From second witness: the suspect was at the gym between 4 PM and 5 PM
9. Cross-reference the person IDs of the suspects in the `person` table

```sql
select * from person
-- 7. from first witness
where id in (select person_id from get_fit_now_member where id like '%48Z%')
and license_id in (select id from drivers_license where plate_number like '%H42W%')
-- 8. from second witness
and id in (
    select person_id from get_fit_now_member
    where id in (
        select membership_id from get_fit_now_check_in
        where check_in_time <= 1700
        and check_out_time >= 1600
        )
    )
```

Common Table Expressions (CTE)

Temporary **named result set that you can reference within another sql statement**

`WITH` **cte_name** `AS` **(subquery)**

```
-- Define the CTE
with suspect_gym as (
    select person_id from get_fit_now_member where id like '48Z%'
)

-- You can reference the CTE like a regular table
select * from person where id in (select * from suspect_gym);
```

Using CTE and subquery to Find the Killer

1️⃣ Define multiple CTEs for each clue

`WITH` cte_name1 `AS` (subquery1), cte_name2 `AS` (subquery2), ...

```
with
suspect_gym as (select person_id from get_fit_now_member where id like '48Z%'),
suspect_license as (select id from drivers_license where plate_number like '%H42W%'),
suspect_gym_checkin_time as (
    select person_id from get_fit_now_member
    where id in (
        select membership_id from get_fit_now_check_in
        where check_in_time <= 1700
        and check_out_time >= 1600
        )
    )
```

Using CTE and subquery to Find the Killer

2️⃣ Use the CTEs in the main query

```
select * from person
where id in (select * from suspect_gym)
and license_id in (select * from suspect_license)
and id in (select * from suspect_gym_checkin_time)
```

‼️ CTE can be referenced only once

```
with suspect_gym as (
    select person_id from get_fit_now_member where id like '48Z%'
)

select * from suspect_gym;
select * from suspect_gym; -- ERROR: suspect_gym can be used only once
```

Clue #2

> I know that she attended the SQL Symphony Concert 3 times in December 2017.

Select all attendance of SQL Symphony Concert in 2017

Table: `facebook_event_checkin`

Columns: all

Rows:

- SQL Symphony Concert
- December 2017

```
select *
from facebook_event_checkin
where event_name like '%SQL Symphony Concert%'
and date between 20170101 and 20171231
order by person_id
```

Aggregation with functions

- `count()`
- `sum()`
- `avg()`
- `max()`
- `min()`
- `stddev()` : standard deviation
- `variance()` : variance

...

---

https://dev.mysql.com/doc/refman/8.0/en/aggregate-functions.html

`count(*)` to count the number of rows

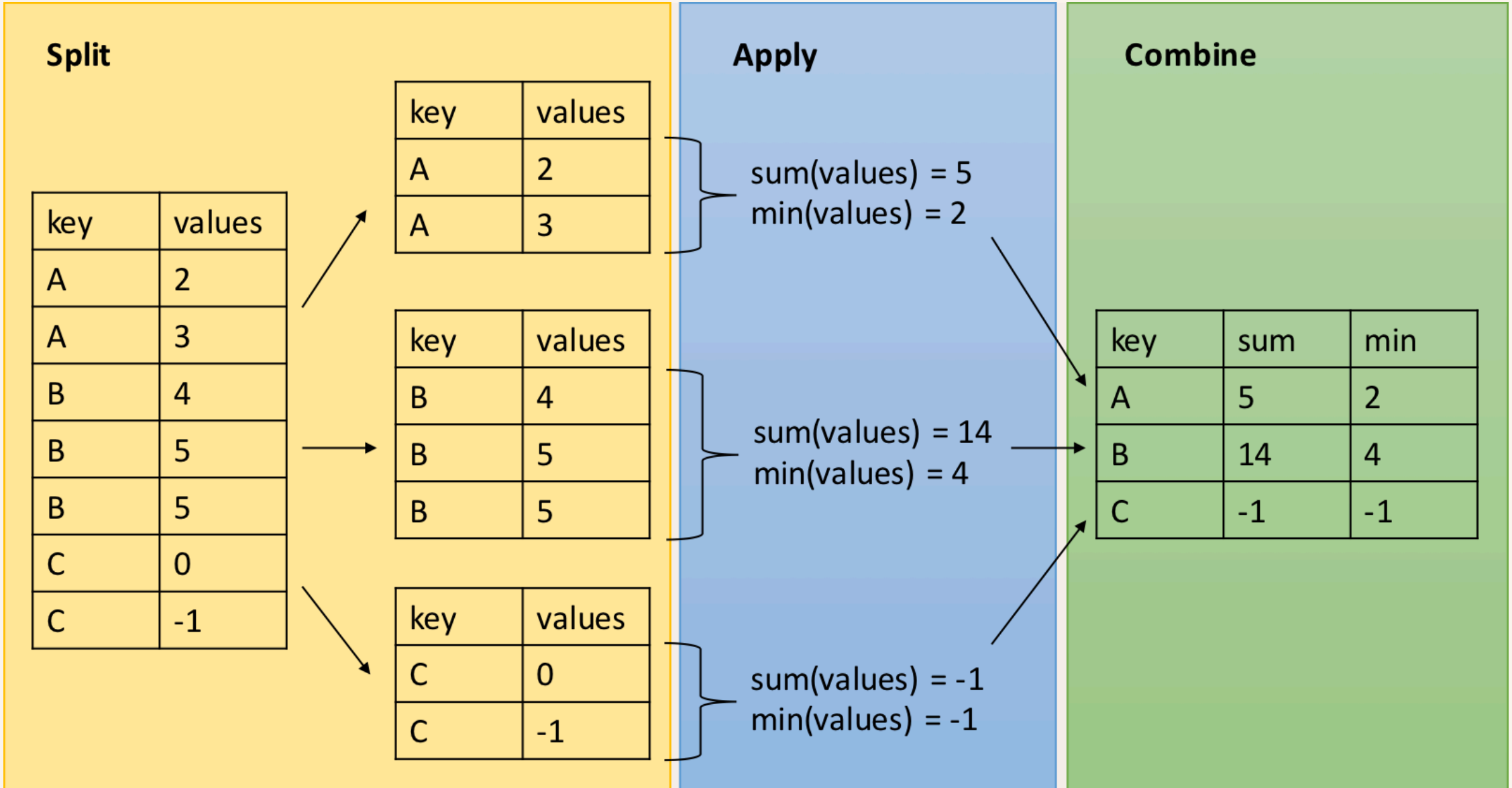
```
select count(*)
from facebook_event_checkin
where event_name like '%SQL Symphony Concert%'
and date between 20170101 and 20171231
order by person_id
```

`select` column `from` table `group by` column

`group by`:

- group rows that have the same values in specified columns
- "find the number of attendance for each person"

Group rows by columns, apply aggregate functions to each group, and combine the results



`GROUP BY` to group rows for each person and aggregate their attendance

```
select person_id, count(*)
from facebook_event_checkin
where event_name like '%SQL Symphony Concert%'
and date between 20170101 and 20171231
group by person_id
```

`HAVING` to filter groups

`WHERE` : filter rows before grouping

`HAVING` : filter groups after grouping

```
select person_id, count(*)
from facebook_event_checkin

-- filter rows before grouping
where event_name like '%SQL Symphony Concert%'
and date between 20170101 and 20171231
group by person_id

-- filter groups after grouping
having count(*) = 3
```

Alias `AS`

```
select person_id, count(*) as num_attendance, count(*) num_attendance2
from facebook_event_checkin
where event_name like '%SQL Symphony Concert%'
and date between 20170101 and 20171231
group by person_id
-- Use alias instead of having count(*) = 3
having num_attendance = 3
```

`ORDER BY` works with aggregate functions

Q: Who attended SQL Symphony Concert the most in 2017?

```
select person_id, count(*) as num_attendance
from facebook_event_checkin
where event_name like '%SQL Symphony Concert%'
and date between 20170101 and 20171231
group by person_id
order by num_attendance desc
```

`DISTINCT` to remove duplicates

Q: How many times SQL Symphony Concert was held in 2017?

➡️ Not how many people attended, but how many unique dates it was held on.

```
select count(distinct date)
from facebook_event_checkin
where event_name like '%SQL Symphony Concert%'
and date between 20170101 and 20171231
```

💻 From Facebook Event Checkin table

1. When was the first event recorded?
2. how many people attended each event?
3. Who attended the most events in 2017?

Putting it All Together

1. Clue #1: People that match the description (`license_id` from `drivers_license`)
2. Clue #2: People that attended SQL Symphony Concert 3 times in 2017 (`person_id` from `facebook_event_checkin`)

Using Subquery

```
select *
from person
where license_id in (
    select id
    from drivers_license
    where hair_color = 'red'
    and car_make = 'Tesla'
    and car_model = 'Model S'
    and height between 65 and 67
)
and id in (
    select person_id
    from facebook_event_checkin
    where event_name like '%SQL Symphony Concert%'
    and date between 20170101 and 20171231
    group by person_id
    having count(*)=3
)
```

Using Subquery and CTE

```
with
suspect_license as (
    select id
    from drivers_license
    where hair_color = 'red'
    and car_make = 'Tesla'
    and car_model = 'Model S'
    and height between 65 and 67
),
suspect_event as (
    select person_id
    from facebook_event_checkin
    where event_name like '%SQL Symphony Concert%'
    and date between 20170101 and 20171231
    group by person_id
    having count(*)=3
)

select *
from person
where license_id in (select * from suspect_license)
and id in (select * from suspect_event)
```

Using `JOIN`

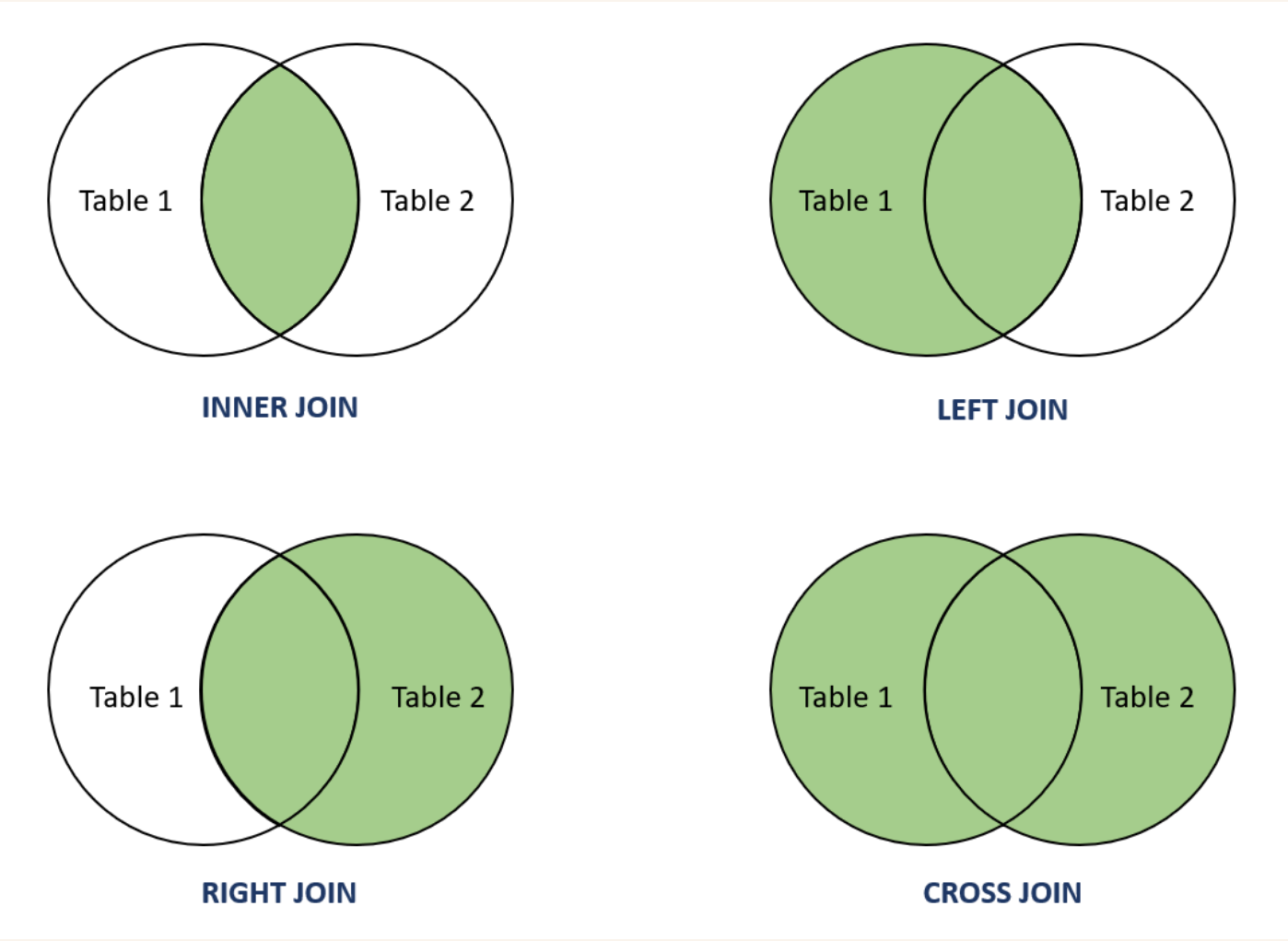


Table 1
Table 2
INNER JOIN
Table 1
Table 2
LEFT JOIN
Table 1
Table 2
RIGHT JOIN
Table 1
Table 2
CROSS JOIN

Examples

id	name	license_id
1	Alice	101
2	Bob	102
3	Charlie	103

id	hair_color	car_make
101	red	Tesla
102	blue	Ford
104	black	BMW

```
JOIN ON
person.license_id = drivers_license.id
```

- **Inner Join: 101, 102**
- Left Join: 101, 102, 103
- Right Join: 101, 102, 104
- Cross Join: 3 x 3 = 9 rows

`JOIN ON` (inner join)

- What tables do you want to join?: `person` and `drivers_license`
- What column(s) do you want to join on?: `license_id` from `person` and `id` from `drivers_license`
- `table_name.column_name` : to specify which table the column belongs to

```
select person.id, drivers_license.id
from (
    -- tables to join
    person join drivers_license
    -- join condition
    on person.license_id = drivers_license.id
    )
```

What column(s) do you want to join on?

1. Join `person` and `drivers_license` on?
2. Join `person` and `facebook_event_checkin` on?

Join `person` and `drivers_license`

```
select *
from person
join drivers_license on person.license_id = drivers_license.id
where hair_color = 'red'
and car_make = 'Tesla'
and car_model = 'Model S'
and height between 65 and 67
```

Putting it All Together with `JOIN` and Subquery

Join `person` and `drivers_license`, and use subquery for `facebook_event_checkin`

```
select *
from person
join drivers_license on person.license_id = drivers_license.id
where hair_color = "red"
and car_make = "Tesla"
and car_model = "Model S"
and height between 65 and 67
and person.id in (
    select person_id
    from facebook_event_checkin
    where event_name like "%SQL Symphony Concert%"
    and date between 20170101 and 20171231
    group by person_id
    having count(*)=3
    )
```

Other `JOIN` syntaxes

`JOIN USING` or `NATURAL JOIN`:

when the columns that you want to join on have the same name in both tables

```
-- Join on license_id column
select * from person join drivers_license using (license_id);
-- Join on all columns with the same name
select * from person natural join drivers_license;
```

`WHERE`:

Joining condition in `WHERE` clause instead of `JOIN` clause

```
select * from person, drivers_license
where person.license_id = drivers_license.id
```

`LEFT JOIN` and `RIGHT JOIN`

`LEFT JOIN`:

Table on the left: all rows

Table on the right: only matching rows

```
select * from person left join get_fit_now_member on person.id = get_fit_now_member.person_id
```

`RIGHT JOIN`:

Table on the left: only matching rows

Table on the right: all rows

```
select * from person right join get_fit_now_member on person.id = get_fit_now_member.person_id
```

Wrap up

- Subquery: query within a query
- Common Table Expressions (CTE): temporary named result set
- Aggregate functions: `count()`, `sum()`, `avg()`, `max()`, `min()`, ...
- `GROUP BY`: group rows by column(s)
- `HAVING`: filter groups after grouping
- `JOIN`: combine data from multiple tables