CODE CHALLENGE: AGGREGATE FUNCTIONS

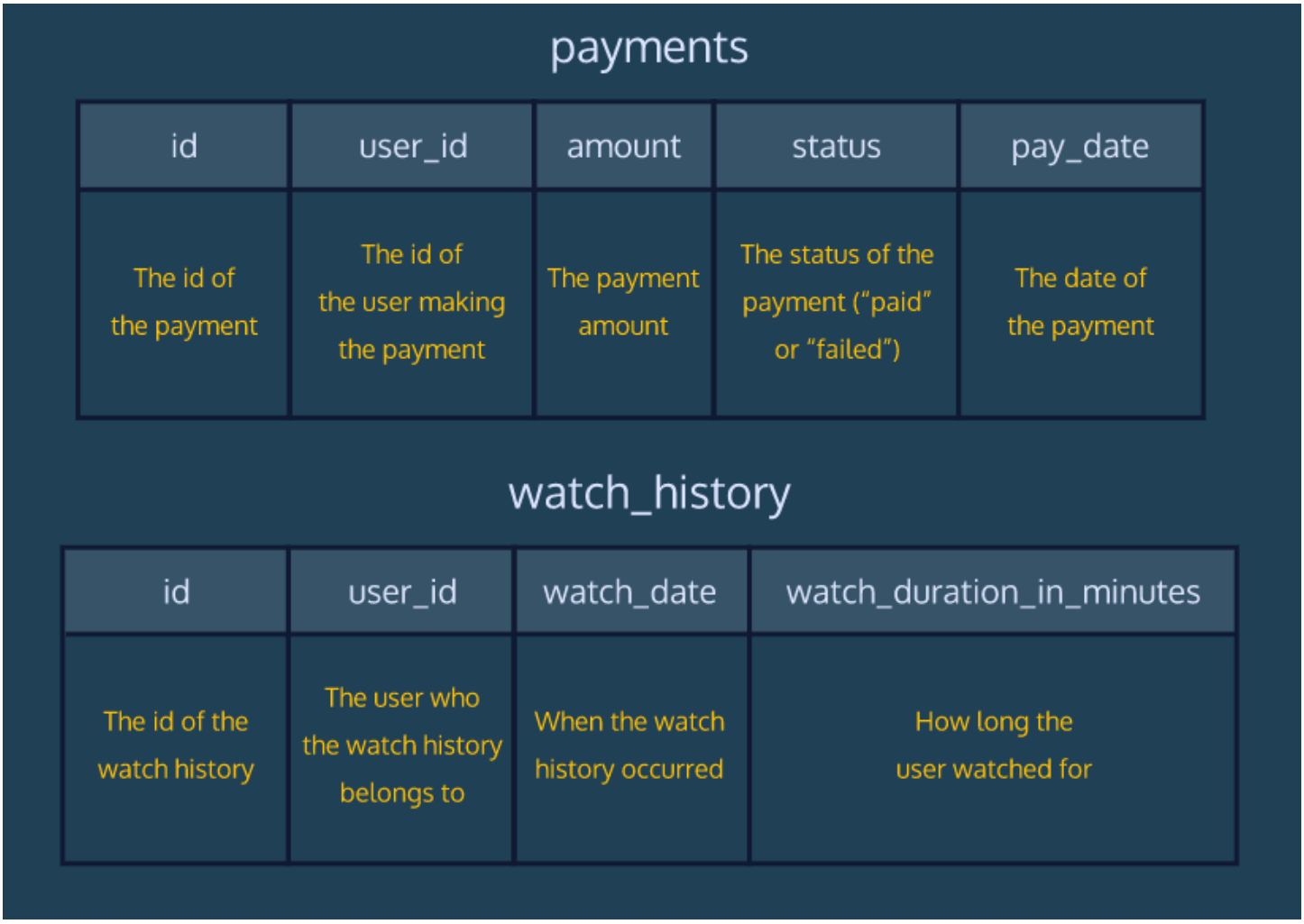
Codeflix Introduction

Welcome to Code Challenge: Aggregate Functions!

A streaming video company, Codeflix, needs your help analyzing their user data. We've imported a portion of their dataset into the SQL workspace for this lesson.

The database contains 3 tables:

- `users` contains the basic account details for each user
- `payments` contains payment details for a 3 month period
- `watch_history` contains watch events for the users



Code Challenge 1

The `users` table has the following columns:

- `id`
- `first_name`
- `last_name`
- `email`
- `password`

Use `COUNT()` and a `LIKE` operator to determine the number of users that have an email ending in '.com'.

```
select count(email)
from users
where email like '%.com';
```

Code Challenge 2

The `users` table has the following columns:

- `id`
- `first_name`
- `last_name`
- `email`
- `password`

What are the most popular first names on Codeflix? Use `COUNT()`, `GROUP BY`, and `ORDER BY` to create a list of first names and occurrences within the `users` table. Order the data so that the most popular names are displayed first.

```
select first_name, count(first_name)
from users
group by 1
order by 2 desc;
```

Code Challenge 3

The `watch_history` table has the following columns:

- `id`
- `user_id`
- `watch_date`
- `watch_duration_in_minutes`

The UX Research team wants to see a distribution of watch durations. They want the result to contain:

- `duration`, which is the watch event duration, rounded to the closest minute
- `count`, the number of watch events falling into this "bucket"

Your result should look like:

duration	count
1.0	9
2.0	21
3.0	19
…	…

Use `COUNT()`, `GROUP BY`, and `ORDER BY` to create this result and order this data by increasing `duration`.

```
select round(watch_duration_in_minutes), count(1)
from watch_history
group by 1
order by 1;
```

Code Challenge 4

The `payments` table has the following columns:

- `id`
- `user_id`
- `amount`
- `status`
- `pay_date`

Find all the users that have successfully made a payment to Codeflix and find their total amount paid. Sort them by their total payments (from high to low).
Use `SUM()`, `WHERE`, `GROUP BY`, and `ORDER BY`.

```
select user_id, sum(amount)
from payments
where status = 'paid'
group by 1
order by 2 desc;
```

Code Challenge 5

The `watch_history` table has the following columns:

- `id`
- `user_id`
- `watch_date`
- `watch_duration_in_minutes`

Generate a table of user ids and total watch duration for users who watched more than 400 minutes of content. Use `SUM()`, `GROUP BY`, and `HAVING` to achieve this.

```
select user_id, sum(watch_duration_in_minutes)
from watch_history
group by 1
having sum(watch_duration_in_minutes) > 400;
```

Code Challenge 6

The `watch_history` table has the following columns:

- `id`
- `user_id`
- `watch_date`
- `watch_duration_in_minutes`

To the nearest minute, how many minutes of content were streamed on Codeflix?

```
select round(sum(watch_duration_in_minutes))
from watch_history;
```

Code Challenge 7

The `payments` table has the following columns:

- `id`
- `user_id`
- `amount`
- `status`
- `pay_date`

Which days in this period did Codeflix collect the most money?
Your result should have two columns, `pay_date` and total `amount`, sorted by the latter descending.
This should only include successful payments (`status = 'paid'`).
Use `SUM()`, `GROUP BY`, and `ORDER BY`.

```
select pay_date, sum(amount)
from payments
where status = 'paid'
group by 1
order by 2 desc;
```

Code Challenge 8

The `payments` table has the following columns:

- `id`
- `user_id`
- `amount`
- `status`
- `pay_date`
-

When users successfully pay Codeflix (`status = 'paid'`), what is the average payment amount?

```
select avg(amount)
from payments
where status = 'paid';
```

Code Challenge 9

The `watch_history` table has the following columns:

- `id`
- `user_id`
- `watch_date`
- `watch_duration_in_minutes`

Of all the events in the `watch_history` table, what is the duration of the longest individual watch event? What is the duration of the shortest?
Use one query and rename the results to `max` and `min`.

```
select max(watch_duration_in_minutes) as longest,
       min(watch_duration_in_minutes) as shortest
from watch_history;
```