perplexity

✔ Query 1

Get the number of unique users who "LIKE"-reacted to videos from channel 352, skipping the first group.

```
SELECT
COUNT(DISTINCT user_likes.user_id) AS no_of_users_reached
FROM
user_likes
INNER JOIN video ON user_likes.video_id = video.video_id
WHERE
video.channel_id = 352
AND user_likes.reaction_type LIKE '%LIKE%'
GROUP BY
user_likes.reaction_type
LIMIT 1 OFFSET 1;
```

✔ Query 2

Count number of videos from "News for you" channel published in 2018:

```
SELECT
COUNT(*) AS no_of_videos
FROM
channel
INNER JOIN video ON channel.channel_id = video.channel_id
WHERE
channel.name = 'News for you'
AND CAST(strftime('%Y', video.published_datetime) AS INT) = 2018;
```

✔ Query 3

Monthly subscriptions to "Taylor Swift" in 2018:

```
SELECT
CAST(strftime('%m', subscribed_datetime) AS INT) AS month_in_2018,
COUNT(*) AS subscribers_per_month
FROM
channel_user
LEFT JOIN channel ON channel_user.channel_id = channel.channel_id
WHERE
channel.name = 'Taylor Swift'
AND CAST(strftime('%Y', subscribed_datetime) AS INT) = 2018
GROUP BY
month_in_2018
ORDER BY
month_in_2018;
```

✅ Query 4
Channels with at least 5 videos in 2018:

```
SELECT
channel.channel_id,
channel.name AS channel_name,
COUNT(video.video_id) AS no_of_videos
FROM
channel
INNER JOIN video ON channel.channel_id = video.channel_id
WHERE
strftime('%Y', video.published_datetime) = '2018'
GROUP BY
channel.channel_id, channel.name
HAVING
COUNT(video.video_id) >= 5
ORDER BY
channel.channel_id ASC;
```

✅ Query 5
User reactions (LIKE/DISLIKE) to channel 366 videos:

```
SELECT
user_likes.user_id,
COUNT(user_likes.video_id) AS no_of_reactions
FROM
user_likes
INNER JOIN video ON user_likes.video_id = video.video_id
WHERE
video.channel_id = 366
AND user_likes.reaction_type IN ('LIKE', 'DISLIKE')
GROUP BY
user_likes.user_id
ORDER BY
no_of_reactions DESC,
user_likes.user_id ASC;
```

✅ Query 6
Videos with views higher than the average:

```
SELECT
name,
no_of_views
FROM
video
WHERE
no_of_views > (
SELECT AVG(no_of_views) FROM video
)
```

```
ORDER BY
name ASC;
```

✅ Query 7
Users who liked videos from "Android Authority" but not video 1005:

```
SELECT DISTINCT user_likes.user_id AS user_id
FROM
video
INNER JOIN channel ON video.channel_id = channel.channel_id
INNER JOIN user_likes ON video.video_id = user_likes.video_id
WHERE
channel.name = 'Android Authority'
AND user_likes.reaction_type = 'LIKE'
AND user_likes.user_id NOT IN (
SELECT user_likes.user_id
FROM user_likes
WHERE video_id = 1005 AND reaction_type = 'LIKE'
)
ORDER BY user_id ASC;
```

✅ Query 8
Top 5 videos belonging to both genre 201 and 202 by views:

```
SELECT
v.name AS video_name,
v.no_of_views
FROM
video v
INNER JOIN video_genre vg1 ON v.video_id = vg1.video_id
INNER JOIN video_genre vg2 ON v.video_id = vg2.video_id
WHERE
vg1.genre_id = 201
AND vg2.genre_id = 202
GROUP BY
v.video_id
ORDER BY
v.no_of_views DESC, video_name ASC
LIMIT 5;
```

✅ Query 9
Peak hour of LIKEs for comedy genre:

```
SELECT
CAST(strftime('%H', user_likes.reacted_at) AS INTEGER) AS hour_of_engagement,
COUNT(*) AS no_of_likes
FROM
user_likes
INNER JOIN video_genre ON user_likes.video_id = video_genre.video_id
INNER JOIN genre ON video_genre.genre_id = genre.genre_id
```

```
WHERE
user_likes.reaction_type = 'LIKE'
AND genre.genre_type = 'COMEDY'
GROUP BY
hour_of_engagement
ORDER BY
no_of_likes DESC
LIMIT 1;
```

✅ Query 10
Users with 5 or more likes on "Tedx" channel videos:

```
SELECT
user_likes.user_id AS active_user_id,
COUNT(*) AS no_of_likes
FROM
user_likes
INNER JOIN video ON user_likes.video_id = video.video_id
INNER JOIN channel ON video.channel_id = channel.channel_id
WHERE
user_likes.reaction_type = 'LIKE'
AND channel.name = 'Tedx'
GROUP BY
user_likes.user_id
HAVING
no_of_likes >= 5
ORDER BY
no_of_likes DESC,
active_user_id ASC;
```

✅ Query 11
Users who liked at least 2 videos from channel 352:

```
SELECT
user_likes.user_id AS potential_user_id,
COUNT(*) AS no_of_likes
FROM
video
INNER JOIN user_likes ON video.video_id = user_likes.video_id
WHERE
video.channel_id = 352
AND user_likes.reaction_type = 'LIKE'
GROUP BY
user_likes.user_id
HAVING
no_of_likes >= 2
ORDER BY
no_of_likes DESC,
potential_user_id ASC;
```

✅ Query 12
Top 3 most liked genres (with ≥ 5 likes) in India in 2018:
SELECT
video_genre.genre_id,
COUNT(*) AS no_of_likes
FROM
user
INNER JOIN user_likes ON user.user_id = user_likes.user_id
INNER JOIN video_genre ON user_likes.video_id = video_genre.video_id
WHERE
user.country = 'INDIA'
AND user_likes.reaction_type = 'LIKE'
AND CAST(strftime('%Y', user_likes.reacted_at) AS INT) = 2018
GROUP BY
video_genre.genre_id
HAVING
no_of_likes >= 5
ORDER BY
no_of_likes DESC,
video_genre.genre_id ASC
LIMIT 3;

CORRECT THIS ALL CODES

Here are the corrected versions of all your provided SQL queries with explanations of fixes, where needed:

Query 1

Your Query:

```
SELECT
    COUNT(DISTINCT user_likes.user_id) AS no_of_users_reached
FROM
    user_likes
    INNER JOIN video ON user_likes.video_id = video.video_id
WHERE
    video.channel_id = 352
    AND user_likes.reaction_type LIKE '%LIKE%'
GROUP BY
    user_likes.reaction_type
LIMIT 1 OFFSET 1;
```

Issues:

- Using `GROUP BY user_likes.reaction_type` but only want count of unique users for "LIKE" reaction.
- `LIKE '%LIKE%'` is problematic if reaction types are exact (could be just 'LIKE').

- The use of `LIMIT 1 OFFSET 1` to skip the first group is unusual here and can lead to unintended results.

Fixed Query:
To count unique users who "LIKE"-reacted to channel 352 videos, skipping the first group is unclear. If you want to skip the first reaction type grouping, better handle it differently.

If you want just the count of unique users who reacted with LIKE on that channel:

```
SELECT
    COUNT(DISTINCT user_likes.user_id) AS no_of_users_reached
FROM
    user_likes
    INNER JOIN video ON user_likes.video_id = video.video_id
WHERE
    video.channel_id = 352
    AND user_likes.reaction_type = 'LIKE';
```

If skipping the first group means ignoring the first reaction_type alphabetically, then:

```
SELECT
    COUNT(DISTINCT user_likes.user_id) AS no_of_users_reached
FROM
    user_likes
    INNER JOIN video ON user_likes.video_id = video.video_id
WHERE
    video.channel_id = 352
    AND user_likes.reaction_type IN (
        SELECT DISTINCT reaction_type
        FROM user_likes
        WHERE video_id IN (
            SELECT video_id FROM video WHERE channel_id = 352
        )
        ORDER BY reaction_type
        LIMIT -1 OFFSET 1
    );
```

But this is complex and usually not intended.

Summary correction: Use exact match on 'LIKE' and omit `GROUP BY` and `LIMIT` unless specifically required.

Query 2

Your Query:

```
SELECT
    COUNT(*) AS no_of_videos
FROM
    channel
    INNER JOIN video ON channel.channel_id = video.channel_id
WHERE
```

```
    channel.name = 'News for you'
    AND CAST(strftime('%Y', video.published_datetime) AS INT) = 2018;
```

Comments:

- Correctly filters videos published in 2018 with channel name "News for you".
- Casting `strftime('%Y', ...)` to INT is good.

No changes needed.

Query 3

Your Query:

```
SELECT
    CAST(strftime('%m', subscribed_datetime) AS INT) AS month_in_2018,
    COUNT(*) AS subscribers_per_month
FROM
    channel_user
    LEFT JOIN channel ON channel_user.channel_id = channel.channel_id
WHERE
    channel.name = 'Taylor Swift'
    AND CAST(strftime('%Y', subscribed_datetime) AS INT) = 2018
GROUP BY
    month_in_2018
ORDER BY
    month_in_2018;
```

Comment:

- Using LEFT JOIN is okay but INNER JOIN preferred as channel name filtering is applied.
- Casting month as INT to order months numerically.

Optimized:

```
SELECT
    CAST(strftime('%m', subscribed_datetime) AS INT) AS month_in_2018,
    COUNT(*) AS subscribers_per_month
FROM
    channel_user
    INNER JOIN channel ON channel_user.channel_id = channel.channel_id
WHERE
    channel.name = 'Taylor Swift'
    AND CAST(strftime('%Y', subscribed_datetime) AS INT) = 2018
GROUP BY
    month_in_2018
ORDER BY
    month_in_2018;
```

Query 4

Your Query:

```
SELECT
    channel.channel_id,
    channel.name AS channel_name,
    COUNT(video.video_id) AS no_of_videos
FROM
    channel
    INNER JOIN video ON channel.channel_id = video.channel_id
WHERE
    strftime('%Y', video.published_datetime) = '2018'
GROUP BY
    channel.channel_id, channel.name
HAVING
    COUNT(video.video_id) >= 5
ORDER BY
    channel.channel_id ASC;
```

Comment:

- Correct.

Query 5

Your Query:

```
SELECT
    user_likes.user_id,
    COUNT(user_likes.video_id) AS no_of_reactions
FROM
    user_likes
    INNER JOIN video ON user_likes.video_id = video.video_id
WHERE
    video.channel_id = 366
    AND user_likes.reaction_type IN ('LIKE', 'DISLIKE')
GROUP BY
    user_likes.user_id
ORDER BY
    no_of_reactions DESC,
    user_likes.user_id ASC;
```

Comment:

- Correct.

Query 6

Your Query:

```
SELECT
    name,
    no_of_views
FROM
    video
WHERE
    no_of_views > (
        SELECT AVG(no_of_views) FROM video
    )
ORDER BY
    name ASC;
```

Comment:

- Correct.

Query 7

Your Query:

```
SELECT DISTINCT user_likes.user_id AS user_id
FROM
    video
    INNER JOIN channel ON video.channel_id = channel.channel_id
    INNER JOIN user_likes ON video.video_id = user_likes.video_id
WHERE
    channel.name = 'Android Authority'
    AND user_likes.reaction_type = 'LIKE'
    AND user_likes.user_id NOT IN (
        SELECT user_likes.user_id
        FROM user_likes
        WHERE video_id = 1005 AND reaction_type = 'LIKE'
    )
ORDER BY user_id ASC;
```

Comment:

- Correct.

Query 8

Your Query:

```
SELECT
    v.name AS video_name,
    v.no_of_views
FROM
    video v
```

```
    INNER JOIN video_genre vg1 ON v.video_id = vg1.video_id
    INNER JOIN video_genre vg2 ON v.video_id = vg2.video_id
WHERE
    vg1.genre_id = 201
    AND vg2.genre_id = 202
GROUP BY
    v.video_id
ORDER BY
    v.no_of_views DESC, video_name ASC
LIMIT 5;
```

Issues:

- Use of GROUP BY with columns not in aggregate - in SQLite this is tolerated, in stricter SQL (e.g. MySQL with sql_mode=ONLY_FULL_GROUP_BY) it fails.
- Better to select distinct video_id and video attributes.

Fix: Select DISTINCT or add all selected columns in GROUP BY:

```
SELECT
    v.name AS video_name,
    v.no_of_views
FROM
    video v
    INNER JOIN video_genre vg1 ON v.video_id = vg1.video_id
    INNER JOIN video_genre vg2 ON v.video_id = vg2.video_id
WHERE
    vg1.genre_id = 201
    AND vg2.genre_id = 202
GROUP BY
    v.video_id, v.name, v.no_of_views
ORDER BY
    v.no_of_views DESC, video_name ASC
LIMIT 5;
```

Query 9

Your Query:

```
SELECT
    CAST(strftime('%H', user_likes.reacted_at) AS INTEGER) AS hour_of_engagement,
    COUNT(*) AS no_of_likes
FROM
    user_likes
    INNER JOIN video_genre ON user_likes.video_id = video_genre.video_id
    INNER JOIN genre ON video_genre.genre_id = genre.genre_id
WHERE
    user_likes.reaction_type = 'LIKE'
    AND genre.genre_type = 'COMEDY'
GROUP BY
    hour_of_engagement
ORDER BY
```

```
    no_of_likes DESC
LIMIT 1;
```

Comment:

- Correct.

Query 10

Your Query:

```
SELECT
    user_likes.user_id AS active_user_id,
    COUNT(*) AS no_of_likes
FROM
    user_likes
    INNER JOIN video ON user_likes.video_id = video.video_id
    INNER JOIN channel ON video.channel_id = channel.channel_id
WHERE
    user_likes.reaction_type = 'LIKE'
    AND channel.name = 'Tedx'
GROUP BY
    user_likes.user_id
HAVING
    no_of_likes >= 5
ORDER BY
    no_of_likes DESC,
    active_user_id ASC;
```

Comment:

- Correct.

Query 11

Your Query:

```
SELECT
    user_likes.user_id AS potential_user_id,
    COUNT(*) AS no_of_likes
FROM
    video
    INNER JOIN user_likes ON video.video_id = user_likes.video_id
WHERE
    video.channel_id = 352
    AND user_likes.reaction_type = 'LIKE'
GROUP BY
    user_likes.user_id
HAVING
    no_of_likes >= 2
ORDER BY
```

```
    no_of_likes DESC,
    potential_user_id ASC;
```

Comment:

- Correct.

Query 12

Your Query:

```
SELECT
    video_genre.genre_id,
    COUNT(*) AS no_of_likes
FROM
    user
    INNER JOIN user_likes ON user.user_id = user_likes.user_id
    INNER JOIN video_genre ON user_likes.video_id = video_genre.video_id
WHERE
    user.country = 'INDIA'
    AND user_likes.reaction_type = 'LIKE'
    AND CAST(strftime('%Y', user_likes.reacted_at) AS INT) = 2018
GROUP BY
    video_genre.genre_id
HAVING
    no_of_likes >= 5
ORDER BY
    no_of_likes DESC,
    video_genre.genre_id ASC
LIMIT 3;
```

Comment:

- Correct.

Summary

Only major fix is to Query 1 (remove unnecessary group by and limit/offset) and Query 8 (include all selected columns in GROUP BY). For Query 3, switched from LEFT JOIN to INNER JOIN for clarity.

If you want me to provide the full final corrected queries in one place, let me know!