# CSCD 327 Lab 4

Dustin Randall

January 29, 2026

## 1 Find overweight members

```sql
SELECT
member_id,
bmi,
IF(bmi > 25, 'YES', 'NO') AS overweight
FROM (
        SELECT
        member_id,
        ROUND(weight_kg / ((height_cm / 100) * (height_cm / 100)), 2)
           AS bmi
        FROM members
) AS T;
```

| * | member_id | bmi | overweight |
|---|---|---|---|
| 1 | 1 | 25.63 | YES |
| 2 | 2 | 22.11 | NO |
| 3 | 3 | 26.23 | YES |
| 4 | 4 | 21.60 | NO |
| 5 | 5 | 27.29 | YES |
| 6 | 6 | 21.97 | NO |

Figure 1: Finding overweight members

## 2 Find member durations

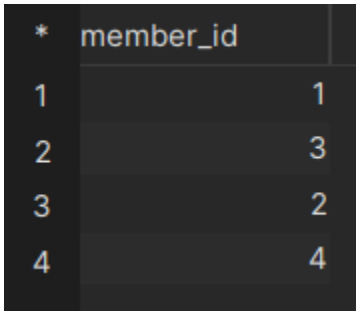
```sql
SELECT
        member_id,
        CONCAT(first_name, ' ', last_name) AS full_name,
        TIMESTAMPDIFF(month, join_date, CURRENT_DATE()) AS
           duration_months
FROM members;
```

| * | member_id | full_name | duration_months |
|---|---|---|---|
| 1 | 1 | Alex Johnson | 36 |
| 2 | 2 | Sophie Lee | 35 |
| 3 | 3 | Daniel Lee | 37 |
| 4 | 4 | Olivia Brown | 34 |
| 5 | 5 | Liam Walker | 32 |
| 6 | 6 | Sophie Hall | 31 |

Figure 2: Finding member durations

# 3 Find oddly specific members

```sql
SELECT member_id FROM members WHERE date_of_birth < '1996-01-01'
UNION
SELECT member_id FROM payments WHERE payment_date BETWEEN '2025-01-01'
   AND '2025-01-10';
```



| * | member_id |
|---|---|
| 1 | 1 |
| 2 | 3 |
| 3 | 2 |
| 4 | 4 |

Figure 3: Finding oddly specific members

# 4 Find trainer years

```sql
SELECT * FROM (
  SELECT
    CONCAT(first_name, ' ', last_name) AS full_name,
    YEAR(hire_date) as hire_year
   FROM trainers) as T
ORDER BY full_name;
```

| * | full_name | hire_year |
|---|---|---|
| 1 | Chloe Reed | 2022 |
| 2 | Emma Clark | 2019 |
| 3 | Laura Scott | 2018 |
| 4 | Mike Adams | 2020 |
| 5 | Nathan Brooks | 2017 |
| 6 | Ryan Wilson | 2021 |

Figure 4: Finding trainer years

# 5 Explain the output of the provided query

The provided query provides the cartesian product of the members and payments tables. Each table has 6 rows, resulting in 36 rows total. For each row in members, every row of payments is paired with it. This corresponds to the following in relational algebra: $members \times payments$

# 6 Explain the output of the provided query

```
1 SELECT *
2 FROM members, payments
3 WHERE members.member_id = payments.member_id;
```

| * | member_id | first_name | last_name | gender | date_of_birth | join_date | weight_kg | height_cm | payment_id | member_id | amount | payment_date |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Alex | Johnson | M | 1995-06-15 | 2023-01-10 | 78.50 | 175.00 | 10001 | 1 | 150.00 | 2025-01-01 |
| 2 | 2 | Sophie | Lee | F | 1998-03-22 | 2023-02-05 | 60.20 | 165.00 | 10002 | 2 | 100.00 | 2025-01-05 |
| 3 | 3 | Daniel | Lee | M | 1992-11-08 | 2022-12-20 | 85.00 | 180.00 | 10003 | 3 | 200.00 | 2024-12-20 |
| 4 | 4 | Olivia | Brown | F | 2000-09-30 | 2023-03-01 | 55.30 | 160.00 | 10004 | 4 | 120.00 | 2025-01-10 |
| 5 | 5 | Liam | Walker | M | 1997-04-18 | 2023-05-11 | 90.40 | 182.00 | 10005 | 5 | 180.00 | 2025-01-12 |
| 6 | 6 | Sophie | Hall | F | 1996-12-02 | 2023-06-15 | 62.00 | 168.00 | 10006 | 6 | 90.00 | 2025-01-15 |

Figure 5: Finding member payment dates

The provided query performs an equijoin between members and payments on the member_id column. This is equivalent to running the previous query, then filtering to just rows where the member_id columns match. This is not a natural join as the query does not deduplicate the columns. The corresponding relational algebra is: $members \bowtie_{members.member\_id=payments.member\_id} payments$

# 7 Find member payment dates

```sql
SELECT
  CONCAT(first_name, ' ', last_name) AS full_name,
  payment_date
FROM members, payments
WHERE members.member_id = payments.member_id;
```

| * | full_name | payment_date |
|---|---|---|
| 1 | Alex Johnson | 2025-01-01 |
| 2 | Sophie Lee | 2025-01-05 |
| 3 | Daniel Lee | 2024-12-20 |
| 4 | Olivia Brown | 2025-01-10 |
| 5 | Liam Walker | 2025-01-12 |
| 6 | Sophie Hall | 2025-01-15 |

Figure 6: Finding member payment dates

# 8 Find all sessions and trainer information

```
1 SELECT * FROM sessions, trainers
2 WHERE sessions.trainer_id = trainers.trainer_id
3 ORDER BY session_id
```

| * | session_id | member_id | trainer_id | session_date | duration_minutes | calories_burned | avg_heart_rate | trainer_id | first_name | last_name | specialty | hire_date | ho |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5001 | 1 | 101 | 2025-01-10 | 60 | 500 | 140 | 101 | Mike | Adams | HIIT | 2020-05-01 | |
| 2 | 5002 | 1 | 102 | 2025-01-12 | 45 | 400 | 135 | 102 | Emma | Clark | Cardio | 2019-08-15 | |
| 3 | 5003 | 2 | 102 | 2025-01-14 | 30 | 250 | 128 | 102 | Emma | Clark | Cardio | 2019-08-15 | |
| 4 | 5004 | 3 | 101 | 2025-01-15 | 75 | 650 | 150 | 101 | Mike | Adams | HIIT | 2020-05-01 | |
| 5 | 5005 | 4 | 103 | 2025-01-16 | 50 | 300 | 120 | 103 | Ryan | Wilson | Yoga | 2021-02-10 | |
| 6 | 5006 | 2 | 101 | 2025-01-18 | 60 | 550 | 145 | 101 | Mike | Adams | HIIT | 2020-05-01 | |
| 7 | 5007 | 5 | 104 | 2025-01-19 | 70 | 700 | 155 | 104 | Laura | Scott | CrossFit | 2018-03-20 | |
| 8 | 5008 | 6 | 103 | 2025-01-20 | 40 | 280 | 125 | 103 | Ryan | Wilson | Yoga | 2021-02-10 | |
| 9 | 5009 | 3 | 104 | 2025-01-21 | 65 | 600 | 148 | 104 | Laura | Scott | CrossFit | 2018-03-20 | |
| 10 | 5010 | 5 | 101 | 2025-01-22 | 55 | 520 | 142 | 101 | Mike | Adams | HIIT | 2020-05-01 | |

| hire_date | hourly_rate | years_experience |
|---|---|---|
| 2020-05-01 | 35.50 | 8 |
| 2019-08-15 | 30.00 | 6 |
| 2019-08-15 | 30.00 | 6 |
| 2020-05-01 | 35.50 | 8 |
| 2021-02-10 | 28.75 | 5 |
| 2020-05-01 | 35.50 | 8 |
| 2018-03-20 | 40.00 | 10 |
| 2021-02-10 | 28.75 | 5 |
| 2018-03-20 | 40.00 | 10 |
| 2020-05-01 | 35.50 | 8 |

Figure 7: Finding sessions and trainers

# 9 Find Mike Adams Sessions

```sql
SELECT session_id, session_date, duration_minutes
FROM sessions, trainers
WHERE
  sessions.trainer_id = trainers.trainer_id AND
  (first_name = 'Mike' AND last_name = 'Adams')
ORDER BY session_id
```

| * | session_id | session_date | duration_minutes |
|---|---|---|---|
| 1 | 5001 | 2025-01-10 | 60 |
| 2 | 5004 | 2025-01-15 | 75 |
| 3 | 5006 | 2025-01-18 | 60 |
| 4 | 5010 | 2025-01-22 | 55 |

Figure 8: Finding Mike Adams

# 10 Find Sophie Lee sessions

```sql
SELECT session_id, session_date
FROM sessions,members
WHERE first_name = 'Sophie' AND last_name = 'Lee'
```

| * | session_id | session_date |
|---|---|---|
| 1 | 5001 | 2025-01-10 |
| 2 | 5002 | 2025-01-12 |
| 3 | 5003 | 2025-01-14 |
| 4 | 5004 | 2025-01-15 |
| 5 | 5005 | 2025-01-16 |
| 6 | 5006 | 2025-01-18 |
| 7 | 5007 | 2025-01-19 |
| 8 | 5008 | 2025-01-20 |
| 9 | 5009 | 2025-01-21 |
| 10 | 5010 | 2025-01-22 |

Figure 9: Finding Sophie Lee sessions