SKILL BUILDER:

Lakshman:

```
SELECT E.EMPLOYEE_NAME, E.CITY
FROM EMPLOYEES E
JOIN COMPANIES C ON E.COMPANY_ID = C.COMPANY_ID;
SELECT E.EMPLOYEE_NAME, C.COMPANY_NAME

FROM EMPLOYEES E
JOIN COMPANIES C ON E.COMPANY_ID = C.COMPANY_ID
WHERE E.SALARY > 15000;
```

Liam:

```
SELECT AVG(S.RATING) AS AVERAGE_RATING, AVG(S.AGE) AS AVERAGE_AGE
FROM SAILORS S
JOIN SAILORS_PRODUCTS SP ON S.SAILOR_ID = SP.SAILOR_ID
JOIN PRODUCTS P ON SP.PRODUCT_ID = P.PRODUCT_ID;

SELECT P.PRODUCT_NAME, P.SELL_PRICE, P.CATEGORY, P.STOCK_COUNT
FROM PRODUCTS P
JOIN SAILORS_PRODUCTS SP ON P.PRODUCT_ID = SP.PRODUCT_ID
JOIN SAILORS S ON SP.SAILOR_ID = S.SAILOR_ID;
```

The Company need to track:

```
SELECT S.SALESMAN_NAME
FROM SALESPEOPLE S
JOIN ORDERS O ON S.SALESMAN_ID = O.SALESMAN_ID
WHERE S.SALARY > 2850;

SELECT C.CLIENT_NAME
FROM CLIENTS C
JOIN ORDERS O ON C.CLIENT_ID = O.CLIENT_ID
WHERE SUBSTR(C.CLIENT_NAME, 2, 1) = 'A';
```

The Company need to manage:

```
SELECT P.PRODUCT_NAME, P.STOCK_COUNT, P.REORDER_LEVEL
FROM PRODUCTS P
WHERE P.STOCK_COUNT < P.REORDER_LEVEL;
```

```
SELECT P.PRODUCT_NAME, SUM(O.QUANTITY) AS TOTAL_QUANTITY_SOLD
FROM PRODUCTS P
JOIN ORDERS O ON P.PRODUCT_ID = O.PRODUCT_ID
GROUP BY P.PRODUCT_NAME;

UPDATE PRODUCTS
SET SELL_PRICE = 950
WHERE PRODUCT_NAME = 'TROUSERS';
```

The company need to manage:

```
SELECT DISTINCT P.PRODUCT_NAME
FROM ORDERS O
JOIN CLIENTS C ON O.CLIENT_ID = C.CLIENT_ID
JOIN PRODUCTS P ON O.PRODUCT_ID = P.PRODUCT_ID
WHERE C.CLIENT_NAME = 'IVAN BAYROSS';

SELECT DISTINCT C.CLIENT_NAME
FROM ORDERS O
JOIN CLIENTS C ON O.CLIENT_ID = C.CLIENT_ID
JOIN PRODUCTS P ON O.PRODUCT_ID = P.PRODUCT_ID
WHERE P.PRODUCT_NAME = 'TROUSERS';
```

The company need to track:

```
SELECT P.PRODUCT_NAME, O.QUANTITY
FROM PRODUCTS P
JOIN ORDERS O ON P.PRODUCT_ID = O.PRODUCT_ID
WHERE O.CLIENT_ID IN ('C00001', 'C00002');

SELECT C.CLIENT_NAME, C.CITY, C.EMAIL, C.PHONE_NUMBER
FROM CLIENTS C
JOIN ORDERS O ON C.CLIENT_ID = O.CLIENT_ID
WHERE O.ORDER_ID = 190001;
```

The company need to calculate:

```
SELECT C.CLIENT_NAME
FROM CLIENTS C
JOIN ORDERS O ON C.CLIENT_ID = O.CLIENT_ID
```

```
JOIN PRODUCTS P ON O.PRODUCT_ID = P.PRODUCT_ID
WHERE (O.QUANTITY * P.SELL_PRICE) >= 10000;

SELECT O.ORDER_ID, SUM(O.QUANTITY) AS TOTAL_QUANTITY
FROM ORDERS O
GROUP BY O.ORDER_ID;
```

Challenge YourSelf:

Sona, a data analyst at a private institute:

```
SELECT C.*, COUNT(E.STUDENT_ID) AS ENROLLMENT_COUNT
FROM COURSES C
LEFT JOIN ENROLLMENTS E ON C.COURSE_ID = E.COURSE_ID
GROUP BY C.COURSE_ID, C.COURSE_NAME, C.COURSE_FEE, C.DURATION;

-- Task 2: Students in courses with more than 3 students
SELECT S.STUDENT_NAME, C.COURSE_NAME
FROM STUDENTS S
JOIN ENROLLMENTS E ON S.STUDENT_ID = E.STUDENT_ID
JOIN COURSES C ON E.COURSE_ID = C.COURSE_ID
WHERE C.COURSE_ID IN (
   SELECT COURSE_ID
   FROM ENROLLMENTS
   GROUP BY COURSE_ID
   HAVING COUNT(*) > 3
);

-- Task 3: Courses and students where fee > 5000
SELECT C.COURSE_NAME, S.STUDENT_NAME, E.ENROLLMENT_DATE
FROM COURSES C
JOIN ENROLLMENTS E ON C.COURSE_ID = E.COURSE_ID
JOIN STUDENTS S ON E.STUDENT_ID = S.STUDENT_ID
WHERE C.COURSE_FEE > 5000;
```

George, a data analyst at a sports analytics company:

```
SELECT p.PLAYER_NAME, g.GAME_NAME
FROM PLAYERS p
JOIN PERFORMANCE pf ON p.PLAYER_ID = pf.PLAYER_ID
JOIN GAMES g ON pf.GAME_ID = g.GAME_ID;

SELECT p.PLAYER_NAME, SUM(pf.SCORE) AS TOTAL_SCORE
FROM PLAYERS p
JOIN PERFORMANCE pf ON p.PLAYER_ID = pf.PLAYER_ID
GROUP BY p.PLAYER_NAME;

SELECT g.GAME_NAME, p.PLAYER_NAME
FROM GAMES g
JOIN PERFORMANCE pf ON g.GAME_ID = pf.GAME_ID
JOIN PLAYERS p ON pf.PLAYER_ID = p.PLAYER_ID
WHERE p.TEAM_NAME = 'Dragons';

SELECT g.GAME_NAME, MAX(pf.SCORE) AS HIGHEST_SCORE
FROM GAMES g
JOIN PERFORMANCE pf ON g.GAME_ID = pf.GAME_ID
GROUP BY g.GAME_NAME;

SELECT DISTINCT p.PLAYER_NAME
FROM PLAYERS p
JOIN PERFORMANCE pf ON p.PLAYER_ID = pf.PLAYER_ID
WHERE pf.PLAY_TIME > 40;
```

Emma, a data analyst at an e-commerce company:

```
SELECT o.ORDER_ID, p.PRODUCT_NAME, o.QUANTITY, o.TOTAL_PRICE
FROM ORDERS o
JOIN PRODUCTS p ON o.PRODUCT_ID = p.PRODUCT_ID;

SELECT p.PRODUCT_NAME
FROM PRODUCTS p
LEFT JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
WHERE o.ORDER_ID = 4;

SELECT p.PRODUCT_NAME, SUM(o.QUANTITY) AS TOTAL_QUANTITY_SOLD
FROM PRODUCTS p
JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
GROUP BY p.PRODUCT_NAME;
```

```
SELECT p.PRODUCT_NAME, o.TOTAL_PRICE
FROM PRODUCTS p
JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
WHERE o.TOTAL_PRICE > 500;

SELECT p.PRODUCT_NAME, p.STOCK, o.QUANTITY
FROM PRODUCTS p
JOIN ORDERS o ON p.PRODUCT_ID = o.PRODUCT_ID
WHERE p.STOCK > o.QUANTITY;
```

Ravi manages a system tracking airline:

```
SELECT F.FLYER_NAME, COALESCE(SUM(M.MILES_EARNED), 0) AS TOTAL_MILES
FROM FLYERS F
LEFT JOIN MILES M ON F.FLYER_ID = M.FLYER_ID
GROUP BY F.FLYER_NAME
ORDER BY F.FLYER_NAME;

SELECT ROUND(AVG(MILES_EARNED), 2) AS AVERAGE_MILES_PER_FLIGHT
FROM MILES;

SELECT F.FLYER_NAME
FROM FLYERS F
LEFT JOIN MILES M ON F.FLYER_ID = M.FLYER_ID
WHERE M.FLYER_ID IS NULL;
```

Practice At Home:

Priya manages a blood donation camp database:

```
SELECT D.DONOR_NAME, SUM(N.UNITS) AS TOTAL_UNITS
FROM DONORS D
LEFT JOIN DONATIONS N ON D.DONOR_ID = N.DONOR_ID
GROUP BY D.DONOR_NAME;

SELECT ROUND(AVG(UNITS), 2) AS AVERAGE_UNITS
FROM DONATIONS;

SELECT D.DONOR_NAME
FROM DONORS D
JOIN DONATIONS N ON D.DONOR_ID = N.DONOR_ID
GROUP BY D.DONOR_NAME
HAVING COUNT(N.DONATION_ID) = 1;
```

Arjun manages a movie rating platform:

```
SELECT M.TITLE, ROUND(AVG(R.RATING), 2) AS AVG_RATING
FROM MOVIES M
JOIN RATINGS R ON M.MOVIE_ID = R.MOVIE_ID
GROUP BY M.TITLE;

SELECT TITLE, AVG_RATING FROM (
  SELECT M.TITLE, ROUND(AVG(R.RATING), 2) AS AVG_RATING
  FROM MOVIES M
  JOIN RATINGS R ON M.MOVIE_ID = R.MOVIE_ID
  GROUP BY M.TITLE
)
WHERE AVG_RATING = (
  SELECT MAX(AVG_RATING) FROM (
    SELECT ROUND(AVG(R.RATING), 2) AS AVG_RATING
    FROM RATINGS R
    GROUP BY R.MOVIE_ID
  )
);

SELECT DISTINCT USER_ID
FROM RATINGS
WHERE RATING > 4.5;
```

Tanvi manages product view tracking on:

```
SELECT P.PRODUCT_NAME, COUNT(V.VIEW_ID) AS TOTAL_VIEWS
FROM PRODUCTS P
LEFT JOIN VIEWS V ON P.PRODUCT_ID = V.PRODUCT_ID
GROUP BY P.PRODUCT_NAME;

SELECT ROUND(AVG(VIEW_COUNT), 2) AS AVERAGE_VIEWS
FROM (
  SELECT PRODUCT_ID, COUNT(*) AS VIEW_COUNT
  FROM VIEWS
  GROUP BY PRODUCT_ID
);

SELECT P.PRODUCT_NAME
FROM PRODUCTS P
LEFT JOIN VIEWS V ON P.PRODUCT_ID = V.PRODUCT_ID
WHERE V.PRODUCT_ID IS NULL;
```

Nisha manages a gym and wants to analyze member:

```
SELECT M.MEMBER_NAME, COUNT(A.ATTENDANCE_ID) AS TOTAL_ATTENDANCE
FROM MEMBERS M
LEFT JOIN ATTENDANCE A ON M.MEMBER_ID = A.MEMBER_ID
GROUP BY M.MEMBER_NAME;

SELECT MAX(ATTEND_COUNT) AS MAX_ATTENDANCE
FROM (
  SELECT MEMBER_ID, COUNT(*) AS ATTEND_COUNT
  FROM ATTENDANCE
  GROUP BY MEMBER_ID
);

SELECT M.MEMBER_NAME
FROM MEMBERS M
LEFT JOIN ATTENDANCE A ON M.MEMBER_ID = A.MEMBER_ID
WHERE A.MEMBER_ID IS NULL;
```

Dinesh maintains a library system:

```
SELECT S.STUDENT_NAME, COUNT(B.BORROW_ID) AS TOTAL_BOOKS_BORROWED
FROM STUDENTS S
LEFT JOIN BORROWINGS B ON S.STUDENT_ID = B.STUDENT_ID
GROUP BY S.STUDENT_NAME;

SELECT ROUND(AVG(BOOK_COUNT), 2) AS AVERAGE_BORROW_COUNT
FROM (
  SELECT STUDENT_ID, COUNT(*) AS BOOK_COUNT
  FROM BORROWINGS
  GROUP BY STUDENT_ID
);

SELECT S.STUDENT_NAME
FROM STUDENTS S
JOIN BORROWINGS B ON S.STUDENT_ID = B.STUDENT_ID
GROUP BY S.STUDENT_NAME
HAVING COUNT(B.BORROW_ID) > 5;
```