1. Stock Market Data: Advanced Window Functions

Table Schema (modeled on S&P 500 Kaggle stock data)

```
CREATE TABLE stocks (
    date DATE,
    open NUMERIC(12,2),
    high NUMERIC(12,2),
    low NUMERIC(12,2),
    close NUMERIC(12,2),
    volume BIGINT,
    ticker VARCHAR(8)
);
```

Sample Data

```
INSERT INTO stocks VALUES
('2025-07-21', 110.5, 112.0, 109.5, 111.0, 1200000, 'AAPL'),
('2025-07-22', 111.0, 113.5, 110.0, 113.2, 1300000, 'AAPL'),
('2025-07-23', 113.2, 115.0, 112.0, 114.8, 1250000, 'AAPL'),
('2025-07-24', 114.8, 117.0, 113.5, 116.5, 1500000, 'AAPL'),
('2025-07-25', 116.5, 118.0, 115.0, 117.5, 1430000, 'AAPL'),
('2025-07-21', 225.0, 228.0, 224.0, 227.0, 950000, 'MSFT'),
('2025-07-22', 227.0, 228.5, 225.5, 226.8, 900000, 'MSFT'),
('2025-07-23', 226.8, 229.0, 225.0, 227.5, 970000, 'MSFT'),
('2025-07-24', 227.5, 231.0, 226.0, 230.2, 990000, 'MSFT'),
('2025-07-25', 230.2, 232.5, 229.5, 231.0, 1010000, 'MSFT');
```

Problem 1: 7-Day Moving Average of “Close” Price per Stock

Requirement: Show each day’s stock “close” price and the 3-day moving average (simulate small data window here for brevity).

```
SELECT
    ticker,
    date,
    close,
    ROUND(AVG(close) OVER (
        PARTITION BY ticker ORDER BY date
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ), 2) AS moving_avg_3day
FROM stocks
ORDER BY ticker, date;
```

Problem 2: Percentage Price Change from Previous Close (Day-Over-Day Return)

```
SELECT
    ticker,
    date,
    close,
    LAG(close) OVER (PARTITION BY ticker ORDER BY date) AS
prev_close,
    ROUND(
        (close - LAG(close) OVER (PARTITION BY ticker ORDER BY
date)) * 100 /
        NULLIF(LAG(close) OVER (PARTITION BY ticker ORDER BY
date), 0), 2
    ) AS pct_change
FROM stocks
ORDER BY ticker, date;
```

- `LAG()` accesses the previous day's close.

Problem 3: Rank Each Day by Volume Traded for Each Ticker (Highest Volume Day in Window)

```
SELECT
    ticker,
    date,
    volume,
    RANK() OVER (PARTITION BY ticker ORDER BY volume DESC) AS
volume_rank
FROM stocks
WHERE date >= '2025-07-21' AND date <= '2025-07-25'
ORDER BY ticker, volume_rank;
```

2. Sports Events Scoring Data: Advanced Window Functions

Table Schema (modeled on Kaggle football player scores)

```
CREATE TABLE player_scores (
    game_id INT,
    player_id INT,
    player_name VARCHAR(100),
    team VARCHAR(30),
    goals INT,
    assists INT,
    yellow_cards INT,
    match_date DATE
);
```

Sample Data

```
INSERT INTO player_scores VALUES
(1001, 201, 'Leo Messi', 'PSG', 2, 1, 0, '2025-07-20'),
(1002, 201, 'Leo Messi', 'PSG', 1, 2, 1, '2025-07-25'),
(1003, 202, 'Kylian Mbappe', 'PSG', 1, 1, 0, '2025-07-20'),
(1004, 202, 'Kylian Mbappe', 'PSG', 3, 0, 0, '2025-07-25'),
(1005, 301, 'Cristiano Ronaldo', 'Al Nassr', 2, 0, 1,
'2025-07-20'),
(1006, 301, 'Cristiano Ronaldo', 'Al Nassr', 1, 1, 0,
'2025-07-25');
```

Problem 4: Running Total Goals Per Player Across Matches

```
SELECT
    player_name,
    match_date,
    goals,
    SUM(goals) OVER (PARTITION BY player_name ORDER BY
match_date) AS running_goals
FROM player_scores
ORDER BY player_name, match_date;
```

Problem 5: Rank Players by Total Assists per Team per Match Day

```
SELECT
    team,
    match_date,
    player_name,
    assists,
    RANK() OVER (PARTITION BY team, match_date ORDER BY assists
DESC) AS assist_rank
FROM player_scores
ORDER BY team, match_date, assist_rank;
```

Problem 6: Get Previous Match’s Goals for Each Player (Using LAG)

```
SELECT
    player_name,
    match_date,
    goals,
    LAG(goals) OVER (PARTITION BY player_name ORDER BY
match_date) AS prev_game_goals
FROM player_scores
ORDER BY player_name, match_date;
```

Problem 7: Find “Best Performance” Game per Player (Max Goals)

```
SELECT *
FROM (
    SELECT
        player_name,
        match_date,
        goals,
        RANK() OVER (PARTITION BY player_name ORDER BY goals
DESC) AS goal_rank
    FROM player_scores
) ranked
WHERE goal_rank = 1;
```

- Shows matches where each player scored their highest goals.