SQL Project Code:

Alden Kallabat

Table Creation:

```sql
CREATE TABLE chess_data (
    game_id VARCHAR(50) PRIMARY KEY,
    rated BOOLEAN,
    turns INTEGER,
    victory_status VARCHAR(20),
    winner VARCHAR(10) CHECK (winner IN ('White', 'Black', 'Draw')),
    time_increment VARCHAR(10),
    white_id VARCHAR(50),
    white_rating INTEGER,
    black_id VARCHAR(50),
    black_rating INTEGER,
    moves TEXT,
    opening_code VARCHAR(10),
    opening_moves TEXT,
    opening_fullname TEXT,
    opening_shortname TEXT,
    opening_response TEXT,
    opening_variation TEXT
);

```

Exploratory Queries:

A)

```sql
SELECT winner,
    COUNT (winner) AS number_of_wins
FROM chess_data
WHERE rated = 'true'
    AND time_increment = '10+0'
    OR time_increment = '8+0'
    OR time_increment = '5+5'
    OR time_increment = '15+0'
    OR time_increment = '5+8'
    OR time_increment = '15+15'
GROUP BY winner
ORDER BY number_of_wins DESC;
```

B)

```sql
WITH mode_diff AS (
    SELECT time_increment,
        COUNT(*) AS total_games,
        ROUND(AVG(ABS(white_rating - black_rating)), 0) AS avg_rating_gap
    FROM chess_data
    WHERE time_increment
        IN ('15+15', '5+8', '15+0', '5+5', '8+0', '10+0')
    GROUP BY time_increment
)
SELECT *,
    ROUND((SELECT AVG(avg_rating_gap)
    FROM mode_diff), 0) AS overall_avg_rating_gap
FROM mode_diff
ORDER BY avg_rating_gap DESC;
```

1.

```sql
SELECT opening_shortname,
    COUNT(opening_shortname) AS total_games,
    COUNT(CASE WHEN winner = 'White' THEN 1 END) AS white_wins,
    COUNT(CASE WHEN winner = 'Black' THEN 1 END) AS black_wins,
    ROUND(100.0 * COUNT(CASE WHEN winner = 'White' THEN 1 END) / COUNT(*), 0)
        AS white_win_percent,
    ROUND(100.0 * COUNT(CASE WHEN winner = 'Black' THEN 1 END) / COUNT(*), 0)
        AS black_win_percent,
    CASE WHEN opening_shortname IN (
         'Queen''s Gambit',
         'English Opening',
         'Ruy Lopez',
         'Italian Game',
         'King''s Pawn Game',
         'Queen''s Pawn Game'
    ) THEN 'White'
        WHEN opening_shortname IN (
         'Philidor Defense',
         'Scandinavian Defense',
         'Caro-Kann Defense',
         'French Defense',
         'Sicilian Defense'
    ) THEN 'Black'
    ELSE 'Unknown'
    END AS opening_initiator
FROM chess_data
WHERE winner IN ('White', 'Black')
GROUP BY opening_shortname
HAVING COUNT(opening_shortname) >= 500
ORDER BY white_win_percent DESC;
```

2.

```sql
SELECT time_increment,
  COUNT(*) AS total_games,
  ROUND(AVG(white_rating)::numeric, 0) AS avg_white_rating,
  ROUND(AVG(black_rating)::numeric, 0) AS avg_black_rating,
  ROUND(((AVG(white_rating)::numeric + AVG(black_rating)::numeric) / 2), 0)
    AS avg_total_rating
FROM chess_data
WHERE rated = 'true'
GROUP BY time_increment
HAVING COUNT(*) > 500
ORDER BY avg_total_rating DESC;
```

3.

```sql
CREATE VIEW rating_diff_percentages AS
WITH rating_diff_bins AS (
    SELECT ROUND((white_rating - black_rating)::numeric / 30) * 30 AS rating_diff,
        winner
    FROM chess_data
    WHERE ABS(white_rating - black_rating) <= 500
),
labeled AS (
    SELECT *,
        CASE WHEN rating_diff > 0 AND winner = 'White' THEN 'higher_win'
            WHEN rating_diff < 0 AND winner = 'Black' THEN 'higher_win'
            WHEN rating_diff = 0 AND winner IN ('White', 'Black') THEN 'higher_win'
            WHEN winner = 'Draw' THEN 'draw'
            ELSE 'lower_win'
        END AS outcome
    FROM rating_diff_bins
    WHERE winner IN ('White', 'Black', 'Draw')
),
aggregated AS (
    SELECT rating_diff,
        COUNT(*) AS total_games,
        SUM(CASE WHEN outcome = 'higher_win' THEN 1 ELSE 0 END) AS higher_wins,
        SUM(CASE WHEN outcome = 'draw' THEN 1 ELSE 0 END) AS draws,
        SUM(CASE WHEN outcome = 'lower_win' THEN 1 ELSE 0 END) AS lower_wins
    FROM labeled
    GROUP BY rating_diff
)
SELECT rating_diff, total_games,
    ROUND(100.0 * higher_wins / total_games, 2) AS higher_win_percent,
    ROUND(100.0 * draws / total_games, 2) AS draw_percent,
    ROUND(100.0 * lower_wins / total_games, 2) AS lower_win_percent
FROM aggregated
WHERE rating_diff <> 0
ORDER BY rating_diff;
```

3.1.

```sql
WITH base AS (
    SELECT rating_diff, total_games, higher_win_percent, draw_percent, lower_win_percent,
        1.0 / (1 + POWER(10, rating_diff::numeric / 400)) AS e
    FROM rating_diff_percentages
)
SELECT rating_diff AS opponent_rating_diff, total_games, higher_win_percent, draw_percent, lower_win_percent,

    -- MY expected win percent
    ROUND(e * 100, 0) AS expected_win_percent,

    -- Expected win percent for the lower-rated player
    ROUND(
        CASE WHEN rating_diff > 0 THEN e
            WHEN rating_diff < 0 THEN 1 - e
            ELSE 0.5 END * 100, 0
    ) AS lower_expected_win_percentage,

    -- Rating gain/loss per result
    ROUND(20 * (1 - e), 4) AS rgw,  -- win
    ROUND(-20 * e, 4) AS rgl,       -- loss
    ROUND(20 * (0.5 - e), 4) AS rgd, -- draw

    -- Rating change over 100 games (opponent is higher rated)
    CASE WHEN rating_diff > 0 THEN ROUND(20 * (1 - e) * higher_win_percent, 4) END AS rcw,
    CASE WHEN rating_diff > 0 THEN ROUND(-20 * e * lower_win_percent, 4) END AS rcl,
    CASE WHEN rating_diff > 0 THEN ROUND(20 * (0.5 - e) * draw_percent, 4) END AS rcd,

    -- Rating change over 100 games (opponent is lower rated)
    CASE WHEN rating_diff < 0 THEN ROUND(20 * (1 - e) * higher_win_percent, 4) END AS rcw,
    CASE WHEN rating_diff < 0 THEN ROUND(-20 * e * lower_win_percent, 4) END AS rcl,
    CASE WHEN rating_diff < 0 THEN ROUND(20 * (0.5 - e) * draw_percent, 4) END AS rcd,

    -- Final rating change over 100 games
    CASE WHEN rating_diff > 0 THEN ROUND(
            20 * ((1 - e) * lower_win_percent + (-e) * higher_win_percent + (0.5 - e) * draw_percent), 0)
        WHEN rating_diff < 0 THEN ROUND(
            20 * ((1 - e) * higher_win_percent + (-e) * lower_win_percent + (0.5 - e) * draw_percent), 0)
    END AS final_rating_change
FROM base;
```