Competitive Analysis with Market Position

```
WITH category_businesses AS (
  SELECT
    business_id,
    name,
    city,
    stars,
    review_count,
    categories,

    LAG(name) OVER (PARTITION BY city, categories ORDER BY stars DESC) as
better_rated_competitor,
    LAG(stars) OVER (PARTITION BY city, categories ORDER BY stars DESC) as
better_rating,
    LEAD(name) OVER (PARTITION BY city, categories ORDER BY stars DESC) as
lower_rated_competitor,
    LEAD(stars) OVER (PARTITION BY city, categories ORDER BY stars DESC) as
lower_rating,

    stars - AVG(stars) OVER (PARTITION BY city, categories) as rating_diff_from_avg,
    -- Market position
    COUNT(*) OVER (PARTITION BY city, categories) as total_competitors,
    ROW_NUMBER() OVER (PARTITION BY city, categories ORDER BY stars DESC) as
competitive_position
  FROM yelp_db.business
  WHERE categories IS NOT NULL
)
SELECT name,
  city,
  categories,
  stars,
  review_count,
  better_rated_competitor,
  better_rating,
  lower_rated_competitor,
  lower_rating,
  ROUND(rating_diff_from_avg, 2) as rating_vs_category_avg,
  total_competitors,
  competitive_position,
  ROUND((competitive_position * 100.0 / total_competitors), 2) as percentile_in_category
FROM category_businesses
WHERE total_competitors > 5
ORDER BY city, categories, stars DESC;
```

Business Ranking Analysis by City with Multiple Metrics

```
WITH business_metrics AS (
  SELECT
    business_id,
    name,
    city,
    state,
    stars,
    review_count,
    categories,
    -- City-level rankings
    dense_rank() OVER (PARTITION BY city ORDER BY stars DESC, review_count DESC) as
city_rank_by_rating,
    dense_rank() OVER (PARTITION BY city ORDER BY review_count DESC) as
city_rank_by_reviews,
    -- Running averages
    AVG(stars) OVER (PARTITION BY city ORDER BY review_count DESC
      ROWS BETWEEN 5 PRECEDING AND CURRENT ROW) as rolling_avg_rating,
    -- Percentile calculation
    NTILE(100) OVER (PARTITION BY city ORDER BY stars) as rating_percentile
  FROM yelp_db.business
)
SELECT
  name,
  city,
  state,
  stars,
  review_count,
  city_rank_by_rating,
  city_rank_by_reviews,
  ROUND(rolling_avg_rating, 2) as avg_rating_last_6_businesses,
  rating_percentile as city_percentile,
  categories
FROM business_metrics
WHERE city_rank_by_rating <= 5  -- Top 5 in each city
  AND review_count > 10
ORDER BY city, city_rank_by_rating;
```

Geographic Performance Trends with Moving Averages

```
WITH geo_metrics AS (
  SELECT business_id, name,  latitude,    longitude,  stars,
    review_count,
    -- Geographic moving averages
    AVG(stars) OVER (
      ORDER BY latitude, longitude
      ROWS BETWEEN 5 PRECEDING AND 5 FOLLOWING
    ) as area_avg_rating,
    -- Cumulative totals
    SUM(review_count) OVER (
      ORDER BY latitude, longitude
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) as cumulative_reviews,
    -- Dense rank for quartile calculation
    NTILE(4) OVER (ORDER BY stars) as rating_quartile,
    -- First value in group
    FIRST_VALUE(name) OVER (
      PARTITION BY ROUND(latitude, 2)
      ORDER BY stars DESC
    ) as best_in_area
  FROM yelp_db.business
  WHERE latitude IS NOT NULL
  AND longitude IS NOT NULL
)
SELECT
  name,
  ROUND(latitude, 4) as lat,
  ROUND(longitude, 4) as long,
  stars,
  review_count,
  ROUND(area_avg_rating, 2) as moving_avg_rating,
  cumulative_reviews,
  rating_quartile,
  best_in_area,
  CASE
    WHEN stars > area_avg_rating THEN 'Above Average'
    WHEN stars < area_avg_rating THEN 'Below Average'
    ELSE 'Average'
  END as performance_indicator
FROM geo_metrics
WHERE review_count > 10
ORDER BY latitude, longitude;
```

Time-Based Rating Analysis Using Postal Codes

```
WITH postal_metrics AS (
  SELECT
    postal_code,
    COUNT(*) OVER (PARTITION BY postal_code) as businesses_in_area,
    AVG(stars) OVER (PARTITION BY postal_code) as area_avg_rating,
    stars,
    review_count,
    name,
    -- Dense ranking within postal code
    DENSE_RANK() OVER (PARTITION BY postal_code ORDER BY stars DESC) as area_rank,
    -- Running totals
    SUM(review_count) OVER (
      PARTITION BY postal_code
      ORDER BY stars DESC
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) as running_review_count,
    -- Percent rank
    PERCENT_RANK() OVER (PARTITION BY postal_code ORDER BY stars) as percentile
  FROM yelp_db.business
  WHERE postal_code IS NOT NULL
)
SELECT
  postal_code,
  name,
  stars,
  review_count,
  businesses_in_area,
  ROUND(area_avg_rating, 2) as average_area_rating,
  area_rank,
  running_review_count,
  ROUND(percentile * 100, 2) as percentile_score,
  CASE
    WHEN percentile >= 0.75 THEN 'Top 25%'
    WHEN percentile >= 0.50 THEN 'Top 50%'
    WHEN percentile >= 0.25 THEN 'Bottom 50%'
    ELSE 'Bottom 25%'
  END as performance_quartile
FROM postal_metrics
WHERE businesses_in_area > 5 and postal_code != ''
ORDER BY postal_code, area_rank;
```

Business Clustering by Location and Performance

```
WITH business_clusters AS (
  SELECT
    business_id,
    latitude,
    longitude,
    ROUND(CAST(latitude AS DECIMAL(10,2))) as lat_group,
    ROUND(CAST(longitude AS DECIMAL(10,2))) as long_group,
    stars,
    review_count,
    categories
  FROM yelp_db.business
  WHERE latitude IS NOT NULL AND longitude IS NOT NULL
),
cluster_metrics AS (
  SELECT
    lat_group,
    long_group,
    COUNT(*) as businesses_in_cluster,
    AVG(stars) as avg_rating,
    AVG(review_count) as avg_reviews,
    -- Using array_join and array_agg instead of STRING_AGG
    array_join(array_agg(DISTINCT split(categories, ',')[1]), ', ') as main_categories
  FROM business_clusters
  GROUP BY lat_group, long_group
  HAVING COUNT(*) > 3
)
SELECT
  lat_group,
  long_group,
  businesses_in_cluster,
  ROUND(avg_rating, 2) as average_rating,
  ROUND(avg_reviews, 0) as average_reviews,
  main_categories
FROM cluster_metrics
ORDER BY businesses_in_cluster DESC, avg_rating DESC
LIMIT 15;
```