KartRider Analytics - Analytical Queries & Results Documentation

Live Application: https://kartrider.kesug.com

Repository: https://github.com/tyrahappy/kartrider-analytics-database.git

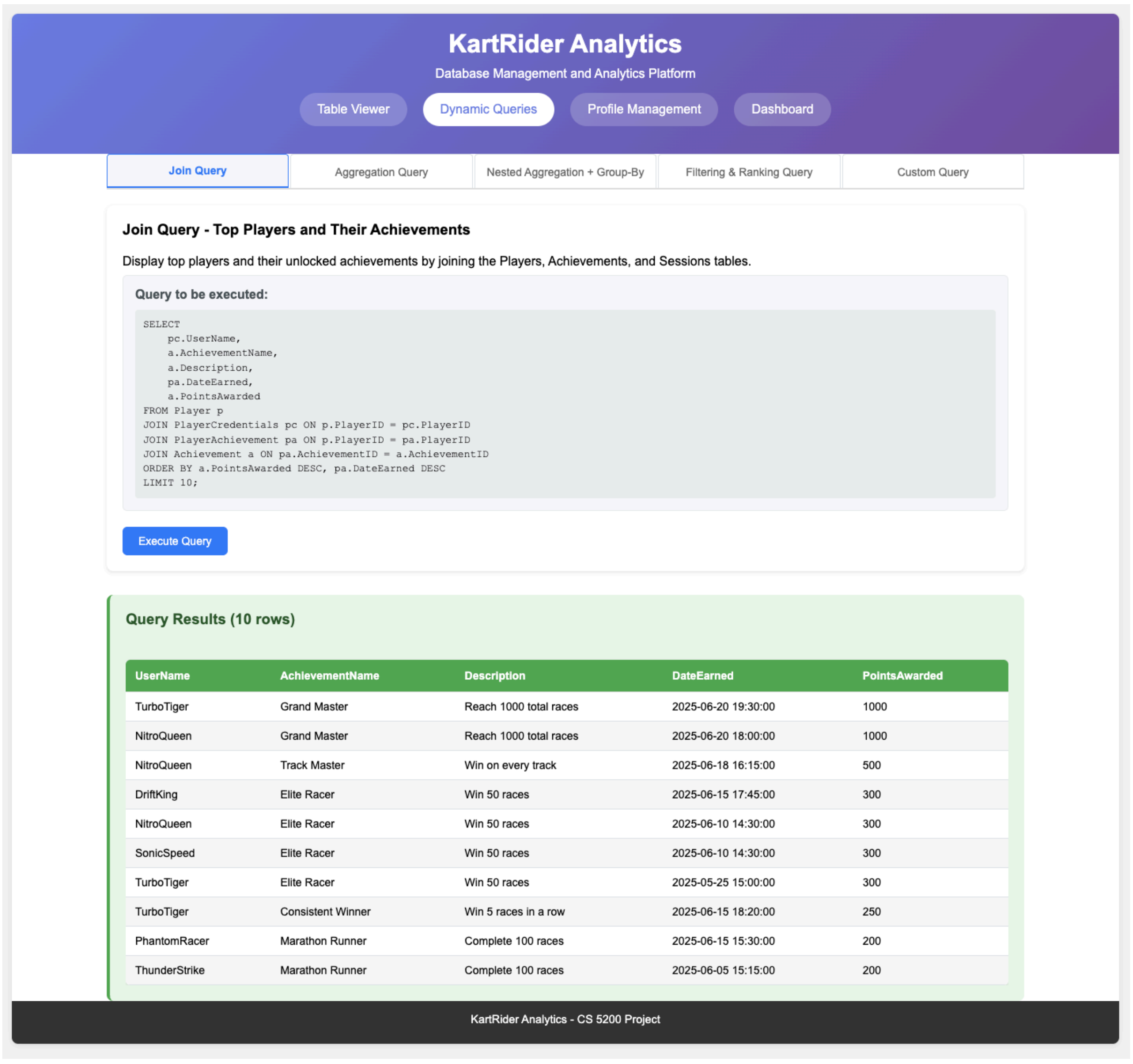
1. Analytical Queries

1.1 Join Query: Top Players and Their Achievements

Requirement: Display top players and their unlocked achievements by joining the Players, Achievements, and Sessions tables.

```
SELECT
     pc.UserName,
     a.AchievementName,
     a.Description,
     pa.DateEarned,
     a.PointsAwarded
FROM Player p
     JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
     JOIN PlayerAchievement pa ON p.PlayerID = pa.PlayerID
     JOIN Achievement a ON pa.AchievementID = a.AchievementID
ORDER BY a.PointsAwarded DESC, pa.DateEarned DESC
LIMIT 10;
```

Query Results:



Join Query - Top Players and Their Achievements

Display top players and their unlocked achievements by joining the Players, Achievements, and Sessions tables.

Query to be executed:

```
SELECT
    pc.UserName,
    a.AchievementName,
    a.Description,
    pa.DateEarned,
    a.PointsAwarded
FROM Player p
JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
JOIN PlayerAchievement pa ON p.PlayerID = pa.PlayerID
JOIN Achievement a ON pa.AchievementID = a.AchievementID
ORDER BY a.PointsAwarded DESC, pa.DateEarned DESC
LIMIT 10;
```

Query Results (10 rows)

UserName	AchievementName	Description	DateEarned	PointsAwarded
TurboTiger	Grand Master	Reach 1000 total races	2025-06-20 19:30:00	1000
NitroQueen	Grand Master	Reach 1000 total races	2025-06-20 18:00:00	1000
NitroQueen	Track Master	Win on every track	2025-06-18 16:15:00	500
DriftKing	Elite Racer	Win 50 races	2025-06-15 17:45:00	300
NitroQueen	Elite Racer	Win 50 races	2025-06-10 14:30:00	300
SonicSpeed	Elite Racer	Win 50 races	2025-06-10 14:30:00	300
TurboTiger	Elite Racer	Win 50 races	2025-05-25 15:00:00	300
TurboTiger	Consistent Winner	Win 5 races in a row	2025-06-15 18:20:00	250
PhantomRacer	Marathon Runner	Complete 100 races	2025-06-15 15:30:00	200
ThunderStrike	Marathon Runner	Complete 100 races	2025-06-05 15:15:00	200

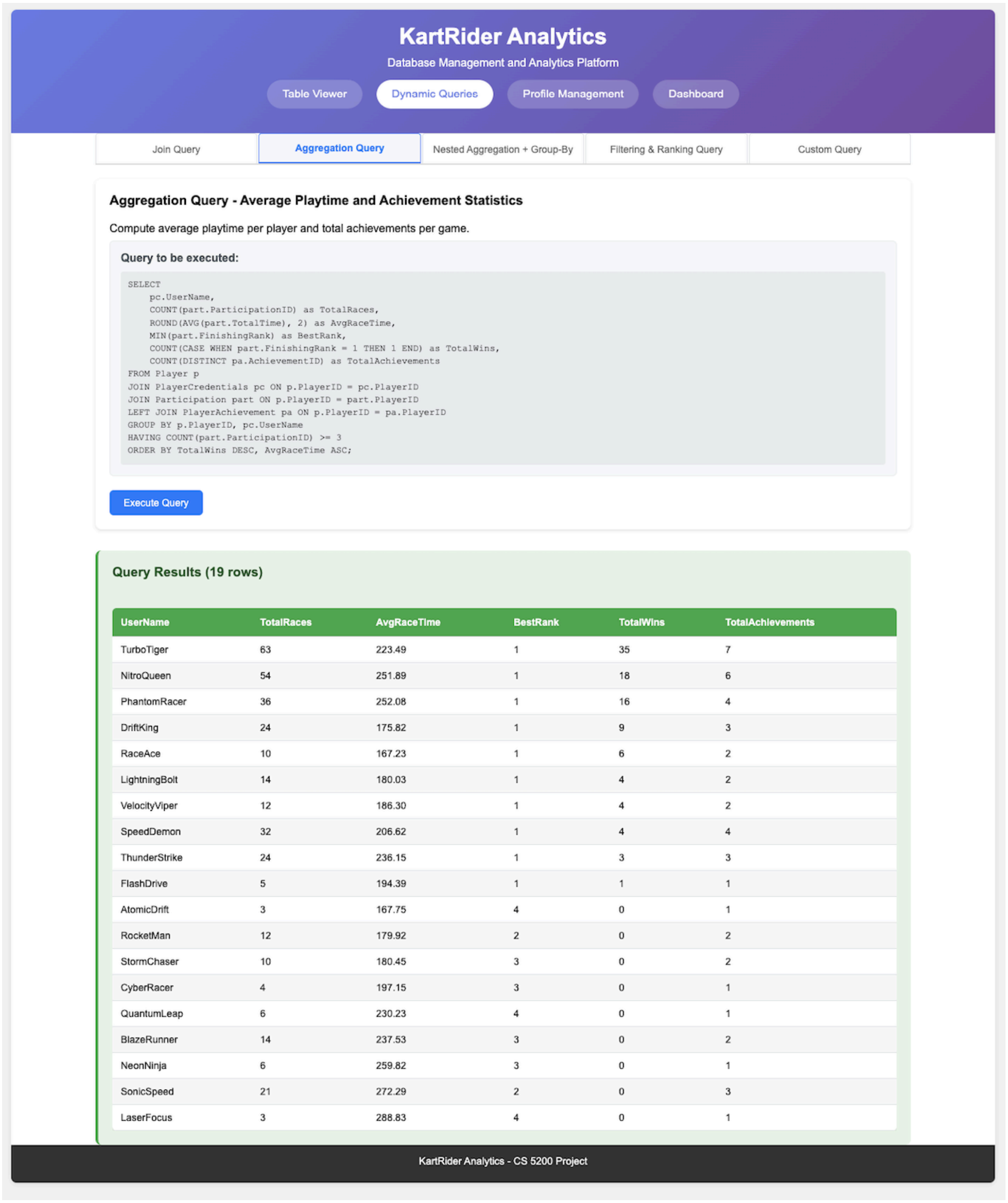
Analysis: This query displays the top players and their achievements based on points and recent activity. Players like TurboTiger and NitroQueen stand out for earning the “Grand Master” achievement (1000 points) by reaching 1000 total races. Others, such as DirtKing and SonicSpeed, achieved “Elite Racer” by winning 50 races, while players like ThunderStrike completed 100 races for the “Marathon Runner” achievement. The results highlight how different achievements reflect various play styles and reward levels.

1.2. Aggregation Query - Average Playtime and Achievement Statistics

Requirement: Compute average playtime per player and total achievements per game.

```
SELECT
	pc.UserName,
	COUNT(part.ParticipationID) as TotalRaces,
	ROUND(AVG(part.TotalTime), 2) as AvgRaceTime,
	MIN(part.FinishingRank) as BestRank, COUNT(CASE WHEN
     part.FinishingRank = 1 THEN 1 END) as TotalWins,
     COUNT(DISTINCT pa.AchievementID) as TotalAchievements
FROM Player p
     JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
     JOIN Participation part ON p.PlayerID = part.PlayerID
     LEFT JOIN PlayerAchievement pa ON p.PlayerID =
pa.PlayerID GROUP BY p.PlayerID, pc.UserName
HAVING COUNT(part.ParticipationID) >= 3
ORDER BY TotalWins DESC, AvgRaceTime ASC;
```

Query Results:


KartRider Analytics
Database Management and Analytics Platform
Table Viewer
Dynamic Queries
Profile Management
Dashboard
Join Query
Aggregation Query
Nested Aggregation + Group-By
Filtering & Ranking Query
Custom Query
Execute Query
KartRider Analytics - CS 5200 Project

Aggregation Query - Average Playtime and Achievement Statistics

Compute average playtime per player and total achievements per game.

Query to be executed:

```
SELECT
    pc.UserName,
    COUNT(part.ParticipationID) as TotalRaces,
    ROUND(AVG(part.TotalTime), 2) as AvgRaceTime,
    MIN(part.FinishingRank) as BestRank,
    COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) as TotalWins,
    COUNT(DISTINCT pa.AchievementID) as TotalAchievements
FROM Player p
JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
JOIN Participation part ON p.PlayerID = part.PlayerID
LEFT JOIN PlayerAchievement pa ON p.PlayerID = pa.PlayerID
GROUP BY p.PlayerID, pc.UserName
HAVING COUNT(part.ParticipationID) >= 3
ORDER BY TotalWins DESC, AvgRaceTime ASC;
```

Query Results (19 rows)

UserName	TotalRaces	AvgRaceTime	BestRank	TotalWins	TotalAchievements
TurboTiger	63	223.49	1	35	7
NitroQueen	54	251.89	1	18	6
PhantomRacer	36	252.08	1	16	4
DriftKing	24	175.82	1	9	3
RaceAce	10	167.23	1	6	2
LightningBolt	14	180.03	1	4	2
VelocityViper	12	186.30	1	4	2
SpeedDemon	32	206.62	1	4	4
ThunderStrike	24	236.15	1	3	3
FlashDrive	5	194.39	1	1	1
AtomicDrift	3	167.75	4	0	1
RocketMan	12	179.92	2	0	2
StormChaser	10	180.45	3	0	2
CyberRacer	4	197.15	3	0	1
QuantumLeap	6	230.23	4	0	1
BlazeRunner	14	237.53	3	0	2
NeonNinja	6	259.82	3	0	1
SonicSpeed	21	272.29	2	0	3
LaserFocus	3	288.83	4	0	1

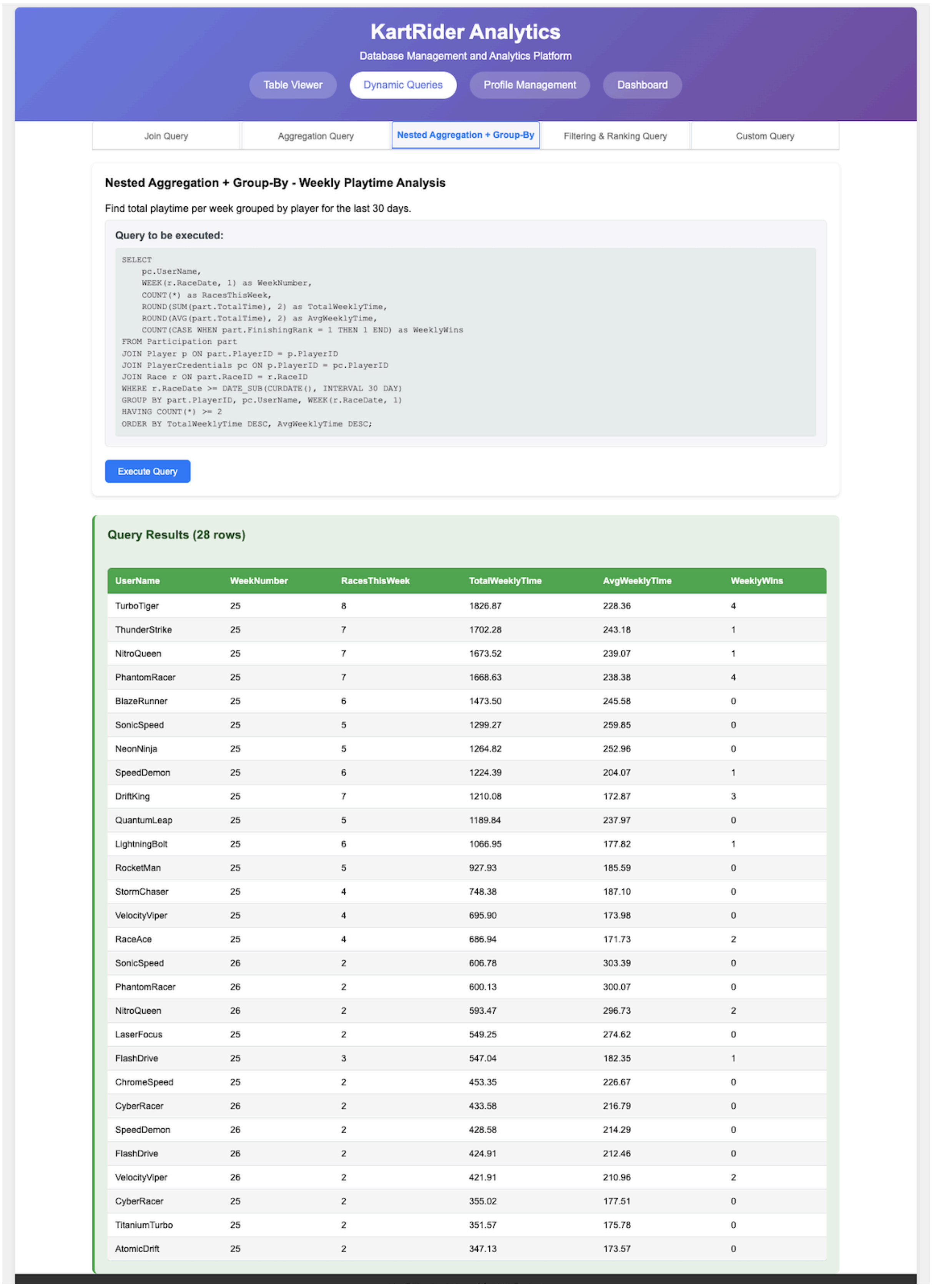
Analysis: This query summarizes player performance by calculating total races, average race time, best rank, total wins, and total achievements. TurboTiger and NitroQueen lead with the highest win counts (35 and 18, respectively) and strong achievement records. PhantomRacer and DirtKing also demonstrate solid performance with notable wins and achievements. The results highlight how players with more wins generally maintain lower (better) average race times and higher achievement counts, reflecting both skill and engagement.

1.3. Nested Aggregation with Group-By - Weekly Playtime Analysis

Requirement: Find total playtime per week grouped by player for the last 30 days.

```
SELECT
    pc.UserName,
    WEEK(r.RaceDate, 1) as WeekNumber,
    COUNT(*) as RacesThisWeek,
    ROUND(SUM(part.TotalTime), 2) as TotalWeeklyTime,
    ROUND(AVG(part.TotalTime), 2) as AvgWeeklyTime,
    COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) as
WeeklyWins
FROM Participation part
     JOIN Player p ON part.PlayerID = p.PlayerID
     JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
     JOIN Race r ON part.RaceID = r.RaceID
WHERE r.RaceDate >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
GROUP BY part.PlayerID, pc.UserName, WEEK(r.RaceDate, 1)
HAVING COUNT(*) >= 2
ORDER BY TotalWeeklyTime DESC, AvgWeeklyTime DESC;
```

Query Results:



Nested Aggregation + Group-By - Weekly Playtime Analysis

Find total playtime per week grouped by player for the last 30 days.

Query to be executed:

```
SELECT
    pc.UserName,
    WEEK(r.RaceDate, 1) as WeekNumber,
    COUNT(*) as RacesThisWeek,
    ROUND(SUM(part.TotalTime), 2) as TotalWeeklyTime,
    ROUND(AVG(part.TotalTime), 2) as AvgWeeklyTime,
    COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) as WeeklyWins
FROM Participation part
JOIN Player p ON part.PlayerID = p.PlayerID
JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
JOIN Race r ON part.RaceID = r.RaceID
WHERE r.RaceDate >= DATE_SUB(CURDATE(), INTERVAL 30 DAY)
GROUP BY part.PlayerID, pc.UserName, WEEK(r.RaceDate, 1)
HAVING COUNT(*) >= 2
ORDER BY TotalWeeklyTime DESC, AvgWeeklyTime DESC;
```

Query Results (28 rows)

UserName	WeekNumber	RacesThisWeek	TotalWeeklyTime	AvgWeeklyTime	WeeklyWins
TurboTiger	25	8	1826.87	228.36	4
ThunderStrike	25	7	1702.28	243.18	1
NitroQueen	25	7	1673.52	239.07	1
PhantomRacer	25	7	1668.63	238.38	4
BlazeRunner	25	6	1473.50	245.58	0
SonicSpeed	25	5	1299.27	259.85	0
NeonNinja	25	5	1264.82	252.96	0
SpeedDemon	25	6	1224.39	204.07	1
DriftKing	25	7	1210.08	172.87	3
QuantumLeap	25	5	1189.84	237.97	0
LightningBolt	25	6	1066.95	177.82	1
RocketMan	25	5	927.93	185.59	0
StormChaser	25	4	748.38	187.10	0
VelocityViper	25	4	695.90	173.98	0
RaceAce	25	4	686.94	171.73	2
SonicSpeed	26	2	606.78	303.39	0
PhantomRacer	26	2	600.13	300.07	0
NitroQueen	26	2	593.47	296.73	2
LaserFocus	25	2	549.25	274.62	0
FlashDrive	25	3	547.04	182.35	1
ChromeSpeed	25	2	453.35	226.67	0
CyberRacer	26	2	433.58	216.79	0
SpeedDemon	26	2	428.58	214.29	0
FlashDrive	26	2	424.91	212.46	0
VelocityViper	26	2	421.91	210.96	2
CyberRacer	25	2	355.02	177.51	0
TitaniumTurbo	25	2	351.57	175.78	0
AtomicDrift	25	2	347.13	173.57	0

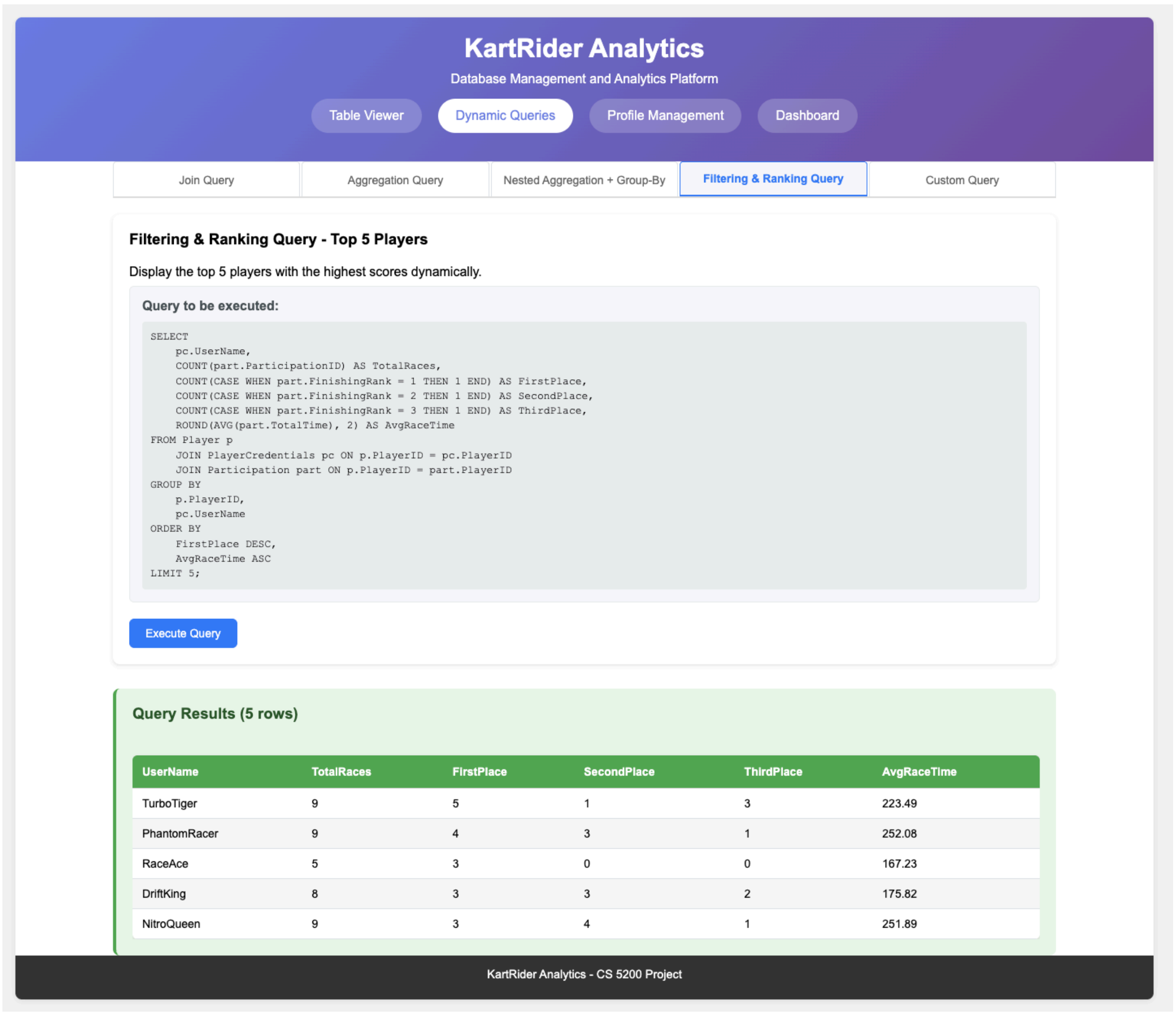
Analysis: This query analyzes weekly player activity over the last 30 days by summarizing total races, total weekly playtime, average playtime, and weekly wins. TurboTiger, ThunderStrike, NitroQueen, and PhantomRacer are the most active players this week, each participating in 7–8 races with the highest total weekly playtime exceeding 1600 minutes. TurboTiger and PhantomRacer also achieved the most weekly wins (4 wins each), indicating both high engagement and strong performance. The results provide clear insights into player commitment and highlight which players are consistently active and successful.

1.4 Filtering & Ranking Query - Top 5 Players

Requirement: Display the top 5 players with the highest scores dynamically.

```
SELECT
    pc.UserName,
    COUNT(part.ParticipationID) AS TotalRaces,
    COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) AS
FirstPlace,
    COUNT(CASE WHEN part.FinishingRank = 2 THEN 1 END) AS
SecondPlace,
    COUNT(CASE WHEN part.FinishingRank = 3 THEN 1 END) AS
ThirdPlace,
    ROUND(AVG(part.TotalTime), 2) AS AvgRaceTime
FROM Player p
    JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
    JOIN Participation part ON p.PlayerID = part.PlayerID
GROUP BY p.PlayerID, pc.UserName
ORDER BY FirstPlace DESC, AvgRaceTime ASC
LIMIT 5;
```

Query Results:


KartRider Analytics
Database Management and Analytics Platform
Table Viewer
Dynamic Queries
Profile Management
Dashboard
Join Query
Aggregation Query
Nested Aggregation + Group-By
Filtering & Ranking Query
Custom Query
Execute Query
KartRider Analytics - CS 5200 Project

Filtering & Ranking Query - Top 5 Players

Display the top 5 players with the highest scores dynamically.

Query to be executed:

```
SELECT
    pc.UserName,
    COUNT(part.ParticipationID) AS TotalRaces,
    COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) AS FirstPlace,
    COUNT(CASE WHEN part.FinishingRank = 2 THEN 1 END) AS SecondPlace,
    COUNT(CASE WHEN part.FinishingRank = 3 THEN 1 END) AS ThirdPlace,
    ROUND(AVG(part.TotalTime), 2) AS AvgRaceTime
FROM Player p
    JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
    JOIN Participation part ON p.PlayerID = part.PlayerID
GROUP BY
    p.PlayerID,
    pc.UserName
ORDER BY
    FirstPlace DESC,
    AvgRaceTime ASC
LIMIT 5;
```

Query Results (5 rows)

UserName	TotalRaces	FirstPlace	SecondPlace	ThirdPlace	AvgRaceTime
TurboTiger	9	5	1	3	223.49
PhantomRacer	9	4	3	1	252.08
RaceAce	5	3	0	0	167.23
DriftKing	8	3	3	2	175.82
NitroQueen	9	3	4	1	251.89

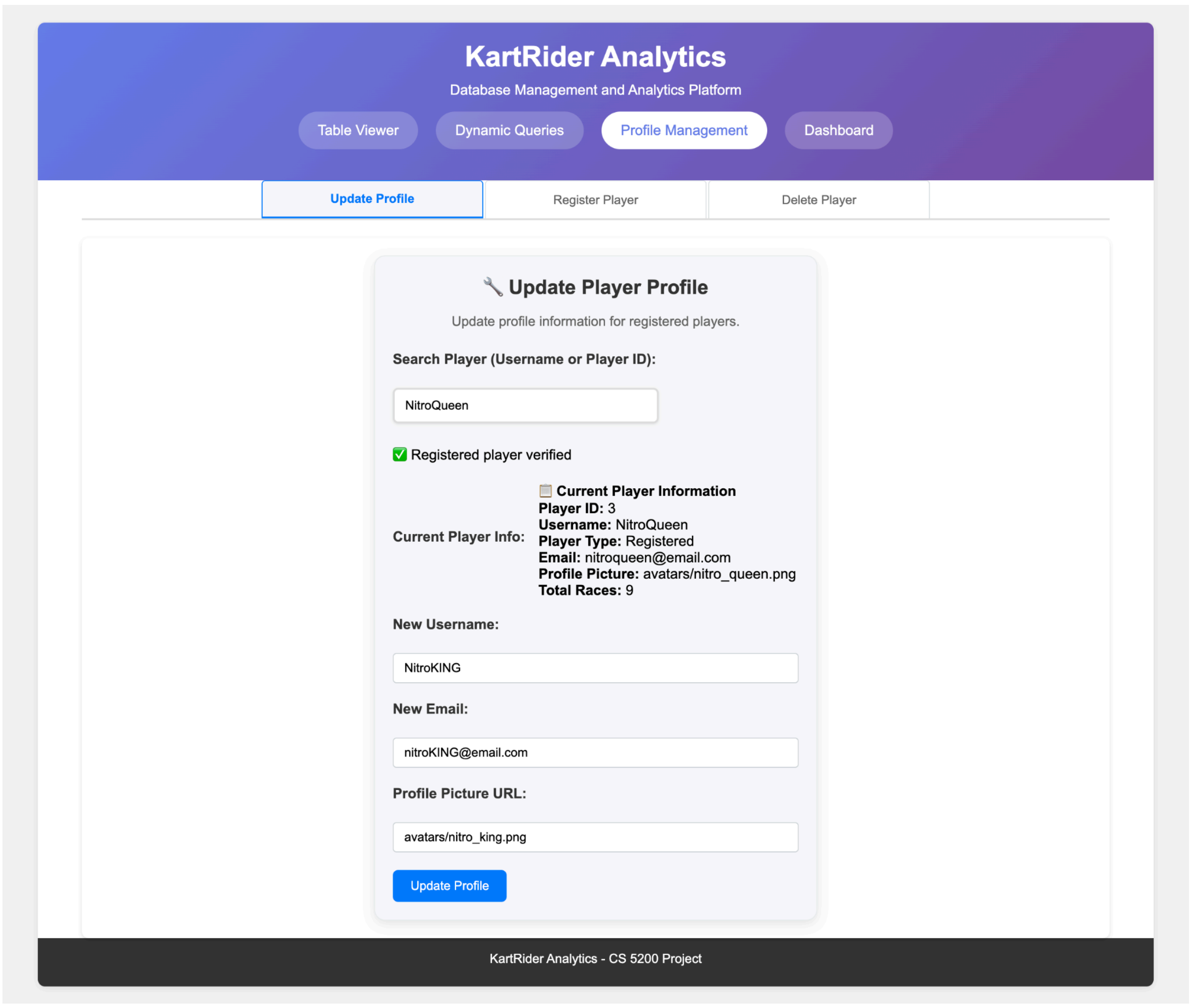
Analysis: This query ranks the top 5 players based on the number of first-place finishes, with average race time used as a tiebreaker. TurboTiger leads with 5 first-place wins, followed closely by PhantomRacer and NitroQueen with 4 each. DriftKing and RaceAce also make the top 5 with strong placements despite having fewer total races. The results highlight not only winning consistency but also racing efficiency, as players with faster average race times are ranked higher when win counts are tied.

1.5 Update Operation - Player Profile Modification

Requirement: Allow users to modify player profile details through web forms.

```sql
-- Update Player Profile Information
UPDATE PlayerCredentials
SET
    UserName = 'SpeedKing2025',
    Email = 'speedking2025@kartrider.com'
WHERE PlayerID = 1;

-- Update Player Statistics
UPDATE Player
SET TotalRaces = (
    SELECT COUNT(*)
    FROM Participation
    WHERE PlayerID = 1
)
WHERE PlayerID = 1;

-- Verification Query
SELECT
    p.PlayerID,
    pc.UserName,
    pc.Email,
    p.TotalRaces,
    rp.ProfilePicture
FROM Player p
     JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
LEFT JOIN RegisteredPlayer rp ON p.PlayerID = rp.PlayerID
WHERE p.PlayerID = 1;
```

Before:



🔧 Update Player Profile

Update profile information for registered players.

Search Player (Username or Player ID): NitroQueen

✅ Registered player verified

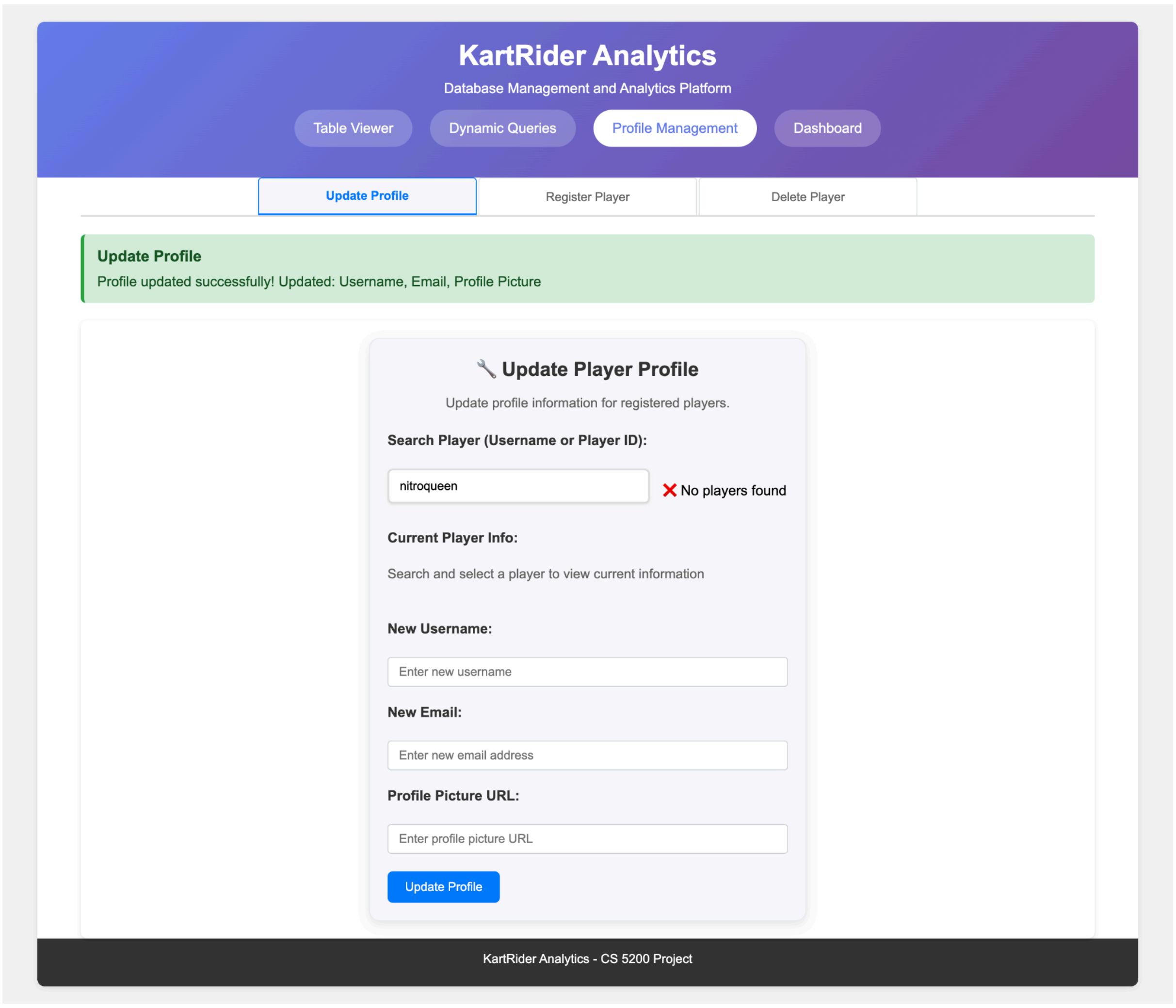
Current Player Info:
📋 **Current Player Information**
Player ID: 3
Username: NitroQueen
Player Type: Registered
Email: nitroqueen@email.com
Profile Picture: avatars/nitro_queen.png
Total Races: 9

New Username: NitroKING

New Email: nitroKING@email.com

Profile Picture URL: avatars/nitro_king.png

After:



KartRider Analytics

Database Management and Analytics Platform

Update Profile

Profile updated successfully! Updated: Username, Email, Profile Picture

🔧 **Update Player Profile**

Update profile information for registered players.

Search Player (Username or Player ID):

nitroqueen ❌ No players found

Current Player Info:

Search and select a player to view current information

New Username:

Enter new username

New Email:

Enter new email address

Profile Picture URL:

Enter profile picture URL

KartRider Analytics - CS 5200 Project

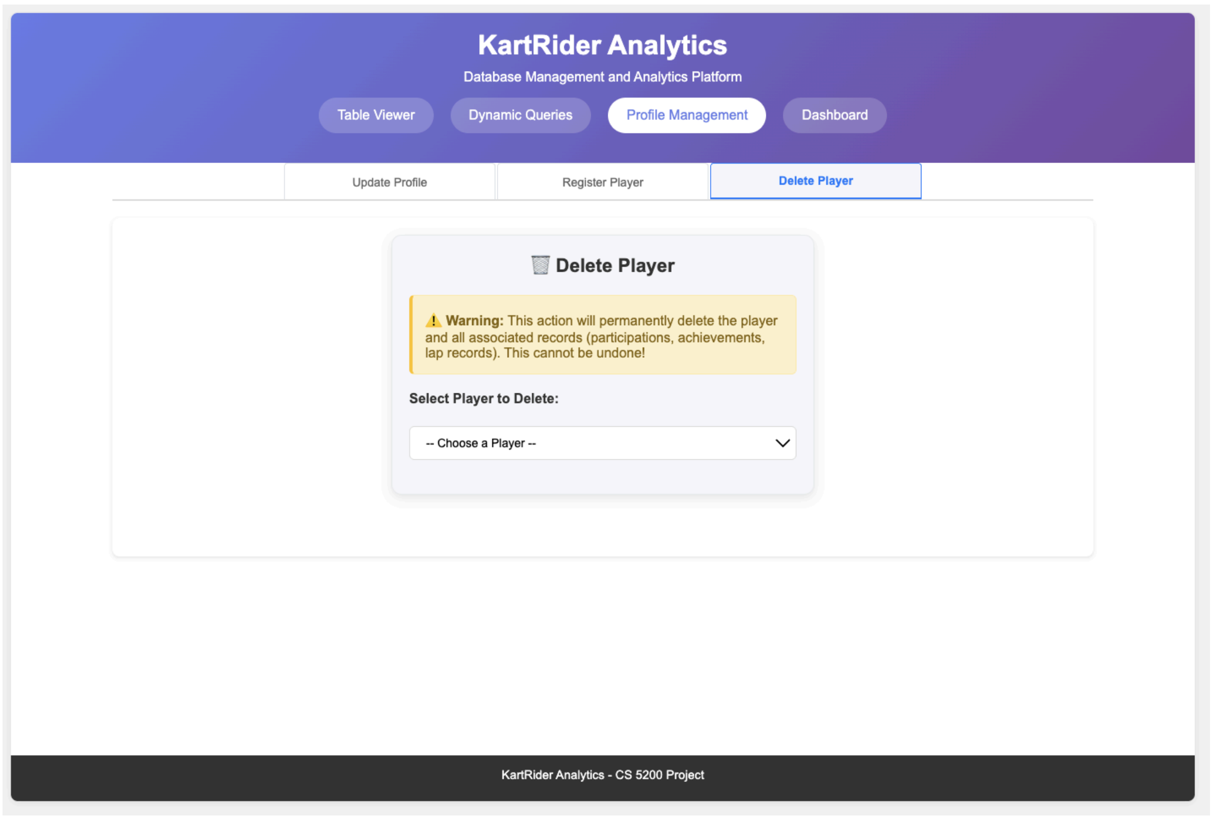
Analysis: The interface shows the transformation from "NitroQueen" to "NitroKING" with updated email and profile picture, proving that the SQL UPDATE statements are properly integrated with the web application's form handling. This implementation showcases proper database transaction management, input validation, and user experience design by providing immediate confirmation messages and maintaining data integrity throughout the update process.

1.6 Delete Operation with Cascade - Data Integrity Management

Requirement: Ensure that deleting a player removes related sessions and achievements automatically.

```
-- Check related records before deletion
SELECT
    'Before Deletion' as Status,
    (SELECT COUNT(*) FROM Player WHERE PlayerID = 30) as
PlayerExists,
    (SELECT COUNT(*) FROM Participation WHERE PlayerID = 30)
as ParticipationRecords,
    (SELECT COUNT(*) FROM PlayerAchievement WHERE PlayerID =
30) as AchievementRecords;

-- Execute cascade delete
DELETE FROM Player WHERE PlayerID = 30;

-- Verify cascade worked
SELECT
    'After Deletion' as Status,
    (SELECT COUNT(*) FROM Player WHERE PlayerID = 30) as
PlayerExists,
    (SELECT COUNT(*) FROM Participation WHERE PlayerID = 30)
as ParticipationRecords,
    (SELECT COUNT(*) FROM PlayerAchievement WHERE PlayerID =
30) as AchievementRecords;
```

Cascade Delete Results:



Delete Player

⚠️ **Warning:** This action will permanently delete the player and all associated records (participations, achievements, lap records). This cannot be undone!

Select Player to Delete:

-- Choose a Player --

KartRider Analytics - CS 5200 Project

Analysis: This operation demonstrates how cascade delete maintains data integrity. When a player (e.g., NitroQueen) is deleted, all related records, including race participation, achievements, and lap records—are automatically removed from the database. The before-and-after checks confirm that no orphaned data remains, ensuring the consistency and reliability of the database.

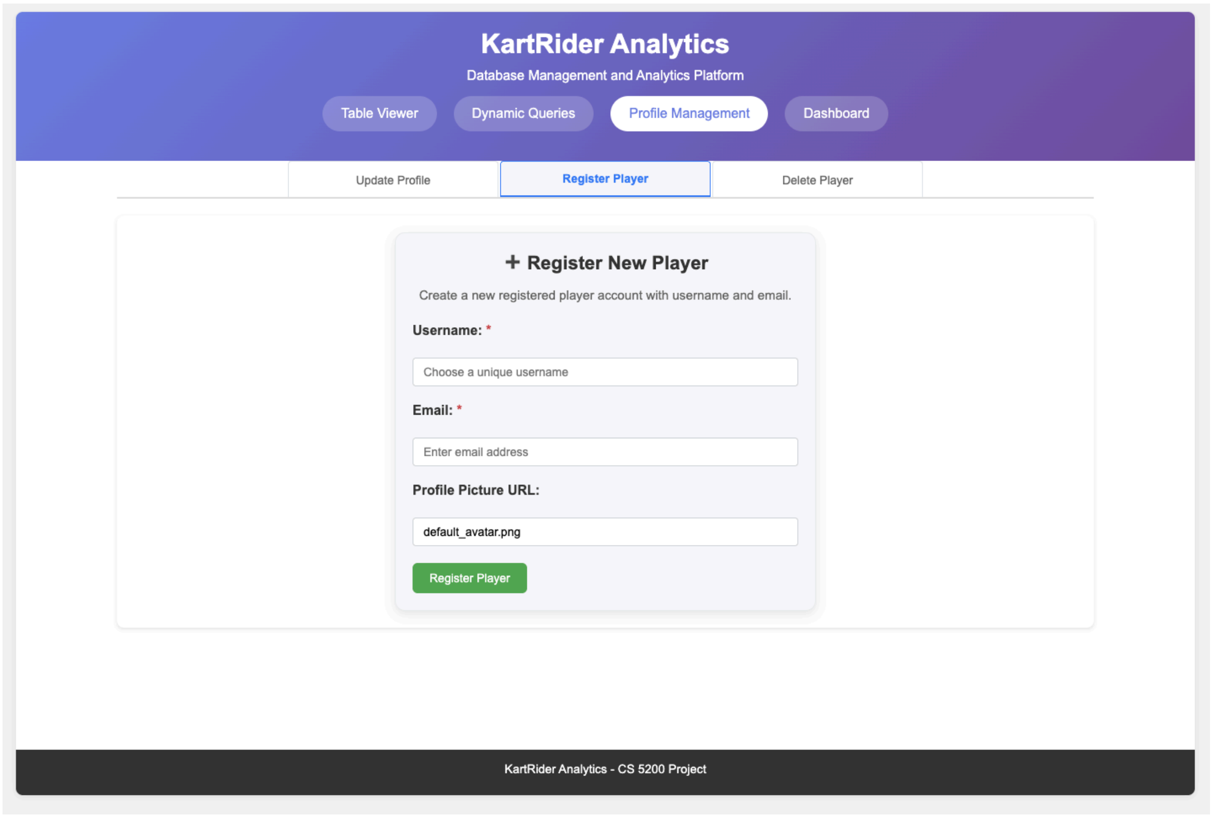
In addition to the required update and delete operations, a register operation is implemented to ensure full CRUD functionality for player profile management.

1.7 Register Operation - Player Profile Creation

Requirement: Allow users to create a new player profile by submitting a username, email, and profile picture through a web form. This operation adds records into both the Player and PlayerCredentials tables.

```
-- Step 1: Username Uniqueness Validation
SELECT PlayerID
FROM PlayerCredentials
WHERE UserName = ?;

-- Step 2: Email Uniqueness Validation
SELECT PlayerID
FROM PlayerCredentials
WHERE Email = ?;

-- Step 3: Create Base Player Record
INSERT INTO Player (TotalRaces)
VALUES (0);

-- Step 5: Create Registered Player Profile
INSERT INTO RegisteredPlayer (PlayerID, ProfilePicture)
VALUES (?, ?);

-- Step 6: Create Player Credentials
INSERT INTO PlayerCredentials (PlayerID, UserName, Email)
VALUES (?, ?, ?);

-- Complete Transaction Example (MySQL)
BEGIN;

-- Validate username doesn't exist
SELECT COUNT(*) as username_count
FROM PlayerCredentials
WHERE UserName = 'NewUsername';

-- Validate email doesn't exist
SELECT COUNT(*) as email_count
FROM PlayerCredentials
```

```sql
WHERE Email = 'new@email.com';

-- If validations pass, create records
INSERT INTO Player (TotalRaces) VALUES (0);
SET @new_player_id = LAST_INSERT_ID();

INSERT INTO RegisteredPlayer (PlayerID, ProfilePicture)
VALUES (@new_player_id, 'default_avatar.png');

INSERT INTO PlayerCredentials (PlayerID, UserName, Email)
VALUES (@new_player_id, 'NewUsername', 'new@email.com');

COMMIT;

-- Verification Query - Check Created Player
SELECT
    p.PlayerID,
    pc.UserName,
    pc.Email,
    p.TotalRaces,
    rp.ProfilePicture,
    'Registered' as PlayerType
FROM Player p
    JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
    JOIN RegisteredPlayer rp ON p.PlayerID = rp.PlayerID
WHERE p.PlayerID = ?;
```



+ Register New Player

Create a new registered player account with username and email.

Username: *

Choose a unique username

Email: *

Enter email address

Profile Picture URL:

default_avatar.png

Analysis: This operation demonstrates the Create component of CRUD. It maintains data integrity by ensuring each player has a corresponding entry in both the Player and PlayerCredentials tables. This allows the user to appear in all subsequent participation, achievement, and profile-related queries, supporting a consistent data structure across the database.

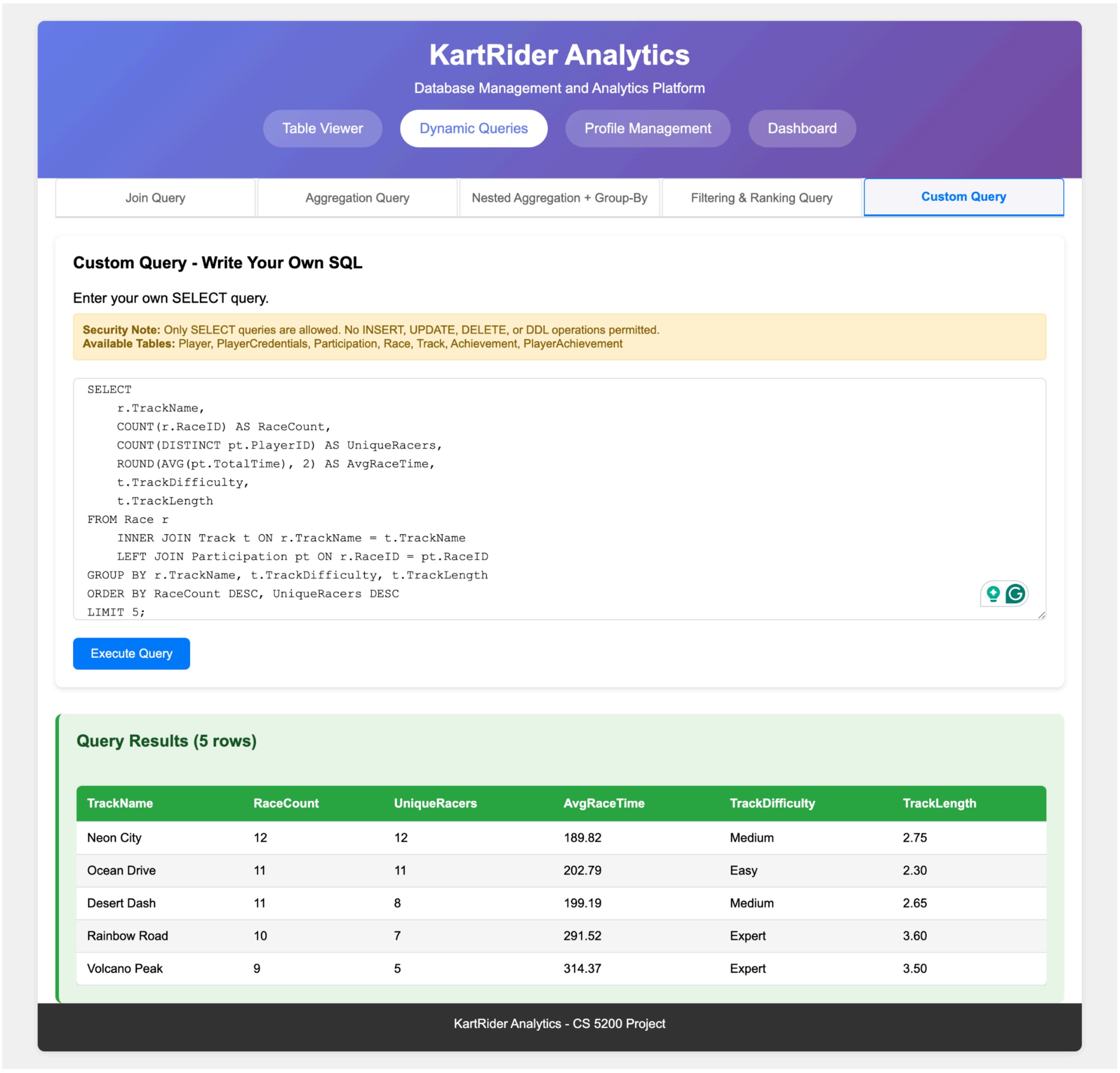
1.8 Custom Query - Write Your Own SQL

Requirement: Allow users to write and execute their own custom **SELECT** SQL queries on the database through the web interface. This feature supports flexible, ad-hoc data exploration beyond predefined queries.

```
-- FILTERING & RANKING QUERY - Top 5 Players by Performance
SELECT
    pc.UserName,
    COUNT(part.ParticipationID) AS TotalRaces,
    COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) AS
FirstPlace,
    COUNT(CASE WHEN part.FinishingRank = 2 THEN 1 END) AS
SecondPlace,
    COUNT(CASE WHEN part.FinishingRank = 3 THEN 1 END) AS
ThirdPlace,
    ROUND(AVG(part.TotalTime), 2) AS AvgRaceTime
FROM Player p
```

```sql
    JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
    JOIN Participation part ON p.PlayerID = part.PlayerID
GROUP BY p.PlayerID, pc.UserName
ORDER BY FirstPlace DESC, AvgRaceTime ASC
LIMIT 5;

-- ADVANCED ANALYSIS - Track Performance Analysis (Default
Example)
SELECT
    t.TrackName,
    COUNT(r.RaceID) AS TotalRaces,
    ROUND(AVG(part.TotalTime), 2) AS AvgRaceTime,
    MIN(part.TotalTime) AS BestTime,
    COUNT(DISTINCT part.PlayerID) AS UniqueRacers,
    t.TrackDifficulty,
    t.TrackLength
FROM Track t
    JOIN Race r ON t.TrackName = r.TrackName
    JOIN Participation part ON r.RaceID = part.RaceID
GROUP BY t.TrackName, t.TrackDifficulty, t.TrackLength
ORDER BY TotalRaces DESC;

-- CUSTOM QUERY EXAMPLES (Additional)

-- Popular Track Analysis
SELECT
    r.TrackName,
    COUNT(r.RaceID) AS RaceCount,
    COUNT(DISTINCT pt.PlayerID) AS UniqueRacers,
    ROUND(AVG(pt.TotalTime), 2) AS AvgRaceTime,
    t.TrackDifficulty,
    t.TrackLength
FROM Race r
    INNER JOIN Track t ON r.TrackName = t.TrackName
    LEFT JOIN Participation pt ON r.RaceID = pt.RaceID
GROUP BY r.TrackName, t.TrackDifficulty, t.TrackLength
ORDER BY RaceCount DESC, UniqueRacers DESC
LIMIT 5;

-- Kart Usage Statistics
SELECT
    k.KartName,
    CASE
        WHEN sk.KartID IS NOT NULL THEN 'Speed'
        WHEN ik.KartID IS NOT NULL THEN 'Item'
        ELSE 'Standard'
    END AS KartType,
    COUNT(pt.ParticipationID) AS UsageCount,
    ROUND(AVG(pt.TotalTime), 2) AS AvgPerformanceTime,
    ROUND(AVG(pt.FinishingRank), 2) AS AvgRank
FROM Participation pt
    INNER JOIN Kart k ON pt.KartID = k.KartID
    LEFT JOIN SpeedKart sk ON k.KartID = sk.KartID
    LEFT JOIN ItemKart ik ON k.KartID = ik.KartID
GROUP BY k.KartID, k.KartName, KartType
ORDER BY UsageCount DESC
LIMIT 10;

-- Player Achievement Progress
SELECT
    pc.UserName,
    COUNT(pa.AchievementID) AS AchievementsEarned,
    SUM(a.PointsAwarded) AS TotalPoints,
    ROUND(COUNT(pa.AchievementID) * 100.0 / (SELECT COUNT(*)
FROM Achievement), 1) AS CompletionRate,
    MAX(pa.DateEarned) AS LastAchievement
FROM Player p
    JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
    LEFT JOIN PlayerAchievement pa ON p.PlayerID = pa.PlayerID
    LEFT JOIN Achievement a ON pa.AchievementID =
a.AchievementID
GROUP BY p.PlayerID, pc.UserName
HAVING COUNT(pa.AchievementID) > 0
ORDER BY TotalPoints DESC, AchievementsEarned DESC;

-- Race Participation Trends by Day
SELECT
    DATE(r.RaceDate) AS RaceDate,
    COUNT(r.RaceID) AS TotalRaces,
    COUNT(pt.ParticipationID) AS TotalParticipations,
    COUNT(DISTINCT pt.PlayerID) AS UniquePlayers,
    ROUND(AVG(pt.TotalTime), 2) AS AvgRaceTime
FROM Race r
    LEFT JOIN Participation pt ON r.RaceID = pt.RaceID
WHERE r.RaceDate >= DATE_SUB(CURDATE(), INTERVAL 14 DAY)
GROUP BY DATE(r.RaceDate)
ORDER BY RaceDate DESC;
```

```
-- Comprehensive Player Performance Summary
SELECT
    pc.UserName,
    p.TotalRaces,
    COUNT(part.ParticipationID) AS ActualRaces,
    COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) AS
Victories,
    ROUND(COUNT(CASE WHEN part.FinishingRank = 1 THEN 1 END) *
100.0 / COUNT(part.ParticipationID), 1) AS WinRate,
    ROUND(AVG(part.TotalTime), 2) AS AvgTime,
    MIN(part.TotalTime) AS BestTime,
    COUNT(DISTINCT part.RaceID) AS UniqueRacesParticipated,
    COUNT(DISTINCT part.KartID) AS DifferentKartsUsed,
    COUNT(DISTINCT pa.AchievementID) AS AchievementsUnlocked
FROM Player p
    JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
    LEFT JOIN Participation part ON p.PlayerID = part.PlayerID
    LEFT JOIN PlayerAchievement pa ON p.PlayerID = pa.PlayerID
GROUP BY p.PlayerID, pc.UserName, p.TotalRaces
HAVING COUNT(part.ParticipationID) > 0
ORDER BY WinRate DESC, AvgTime ASC;
```



Custom Query - Write Your Own SQL

Enter your own SELECT query.

Security Note: Only SELECT queries are allowed. No INSERT, UPDATE, DELETE, or DDL operations permitted.
Available Tables: Player, PlayerCredentials, Participation, Race, Track, Achievement, PlayerAchievement

```
SELECT
    r.TrackName,
    COUNT(r.RaceID) AS RaceCount,
    COUNT(DISTINCT pt.PlayerID) AS UniqueRacers,
    ROUND(AVG(pt.TotalTime), 2) AS AvgRaceTime,
    t.TrackDifficulty,
    t.TrackLength
FROM Race r
    INNER JOIN Track t ON r.TrackName = t.TrackName
    LEFT JOIN Participation pt ON r.RaceID = pt.RaceID
GROUP BY r.TrackName, t.TrackDifficulty, t.TrackLength
ORDER BY RaceCount DESC, UniqueRacers DESC
LIMIT 5;
```

Query Results (5 rows)

TrackName	RaceCount	UniqueRacers	AvgRaceTime	TrackDifficulty	TrackLength
Neon City	12	12	189.82	Medium	2.75
Ocean Drive	11	11	202.79	Easy	2.30
Desert Dash	11	8	199.19	Medium	2.65
Rainbow Road	10	7	291.52	Expert	3.60
Volcano Peak	9	5	314.37	Expert	3.50

Analysis: The Custom Query feature provides users with the flexibility to explore data dynamically without modifying the backend code. It supports only SELECT queries to maintain data integrity and security (no INSERT, UPDATE, DELETE, or DDL operations are permitted). Users can write queries involving tables such as Player, PlayerCredentials, Participation, Race, Track, Achievement, PlayerAchievement.

2. Dashboard Visualization

2.1 Player Statistics Module

The Player Statistics Dashboard provides an overview of player demographics and activity. It displays key metrics such as total players, active players, average races per player, and the active rate in the recent week. The dashboard visualizes the distribution of player types (Guest vs Registered) and race participation levels. It also highlights the top 5 players based on win rate, offering insights into player performance and competitiveness.

Key Features:

- Summary cards: total players, active players, avg races, active rate
- Pie chart: player type distribution (Guest vs Registered)
- Bar chart: race participation distribution
- Leaderboard: top 5 players by win rate
- Filters: by time period and player type

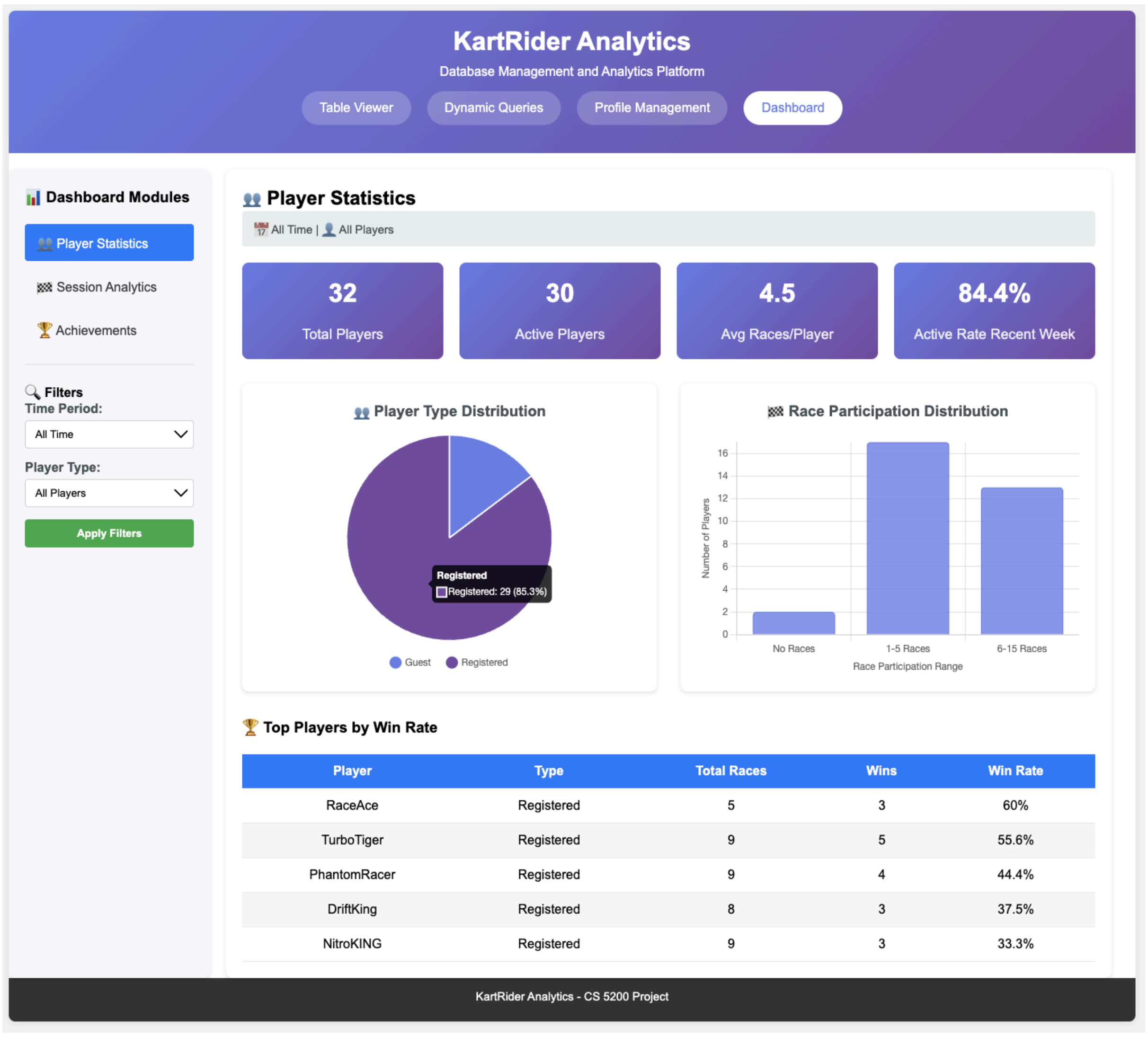
Chart Type: Statistical Dashboard with Pie Chart (Player Type), Bar Chart (Race Participation), and Table (Top Win Rate).

Data Source:

```
-- 1. Summary Statistics (KPI Cards)

-- Total Players Count
SELECT COUNT(*) AS total_players
FROM Player p
LIMIT 1;

-- Active Players Count
SELECT COUNT(DISTINCT PlayerID) AS active_players
FROM Participation
LIMIT 1;

-- Average Races Per Player
SELECT
    ROUND(
        (SELECT COUNT(*) FROM Participation) /
        (SELECT COUNT(DISTINCT PlayerID) FROM Participation),
1
    ) AS avg_races_per_player;

-- Recent Week Activity Rate
SELECT
    ROUND(
        (COUNT(DISTINCT pt.PlayerID) * 100.0 /
         (SELECT COUNT(*) FROM Player)), 1
    ) AS activity_rate_percent
FROM Participation pt
    INNER JOIN Race r ON pt.RaceID = r.RaceID
WHERE r.RaceDate >= DATE_SUB(NOW(), INTERVAL 7 DAY);


-- 2. Player Type Distribution (Pie Chart)

SELECT
```

```
    CASE
        WHEN rp.PlayerID IS NOT NULL THEN 'Registered'
        ELSE 'Guest'
    END AS PlayerType,
    COUNT(*) AS PlayerCount
FROM Player p
    LEFT JOIN RegisteredPlayer rp ON p.PlayerID = rp.PlayerID
GROUP BY (rp.PlayerID IS NOT NULL)
ORDER BY PlayerCount DESC;


-- 3. Race Participation Distribution (Bar Chart)
SELECT
    CASE
        WHEN race_count BETWEEN 1 AND 5 THEN '1-5 Races'
        WHEN race_count BETWEEN 6 AND 15 THEN '6-15 Races'
        WHEN race_count BETWEEN 16 AND 30 THEN '16-30 Races'
        WHEN race_count BETWEEN 31 AND 50 THEN '31-50 Races'
        ELSE '50+ Races'
    END AS ParticipationRange,
    COUNT(*) AS PlayerCount
FROM (
    SELECT
        pt.PlayerID,
        COUNT(pt.RaceID) AS race_count
    FROM Participation pt
        INNER JOIN Race r ON pt.RaceID = r.RaceID
    GROUP BY pt.PlayerID
) player_participation
GROUP BY ParticipationRange
ORDER BY
    CASE ParticipationRange
        WHEN '1-5 Races' THEN 1
        WHEN '6-15 Races' THEN 2
        WHEN '16-30 Races' THEN 3
        WHEN '31-50 Races' THEN 4
        ELSE 5
    END;


-- 4. Top Players by Win Rate (Leaderboard Table)
SELECT
    pc.UserName AS PlayerName,
    CASE
```

```
        WHEN rp.PlayerID IS NOT NULL THEN 'Registered'
        ELSE 'Guest'
    END AS PlayerType,
    COUNT(*) AS TotalRaces,
    SUM(CASE WHEN pt.FinishingRank = 1 THEN 1 ELSE 0 END) AS
Wins,
    ROUND(
        (SUM(CASE WHEN pt.FinishingRank = 1 THEN 1 ELSE 0 END)
/ COUNT(*)) * 100, 1
    ) AS WinRate
FROM Participation pt
    INNER JOIN Player p ON pt.PlayerID = p.PlayerID
    LEFT JOIN PlayerCredentials pc ON p.PlayerID = pc.PlayerID
    LEFT JOIN RegisteredPlayer rp ON p.PlayerID = rp.PlayerID
GROUP BY pt.PlayerID, pc.UserName, rp.PlayerID
HAVING TotalRaces >= 2
ORDER BY WinRate DESC, TotalRaces DESC
LIMIT 5;
```



Player	Type	Total Races	Wins	Win Rate
RaceAce	Registered	5	3	60%
TurboTiger	Registered	9	5	55.6%
PhantomRacer	Registered	9	4	44.4%
DriftKing	Registered	8	3	37.5%
NitroKING	Registered	9	3	33.3%

2.2 Session Analytics Module

This dashboard provides an overview of race participation trends and session data. It highlights total races, average race time, most popular track, and most popular kart. The dashboard also visualizes track difficulty usage, kart usage statistics, and displays recent race activity over the past 7 days. Filters are available to refine data by time period and player type.

Key Features:

- Displays total races, average race time, most popular track, and most popular kart
- Track difficulty usage with a donut chart
- Kart usage statistics with a bar chart
- Recent race activity (Last 7 days) in a data table
- Filters by date range and player type

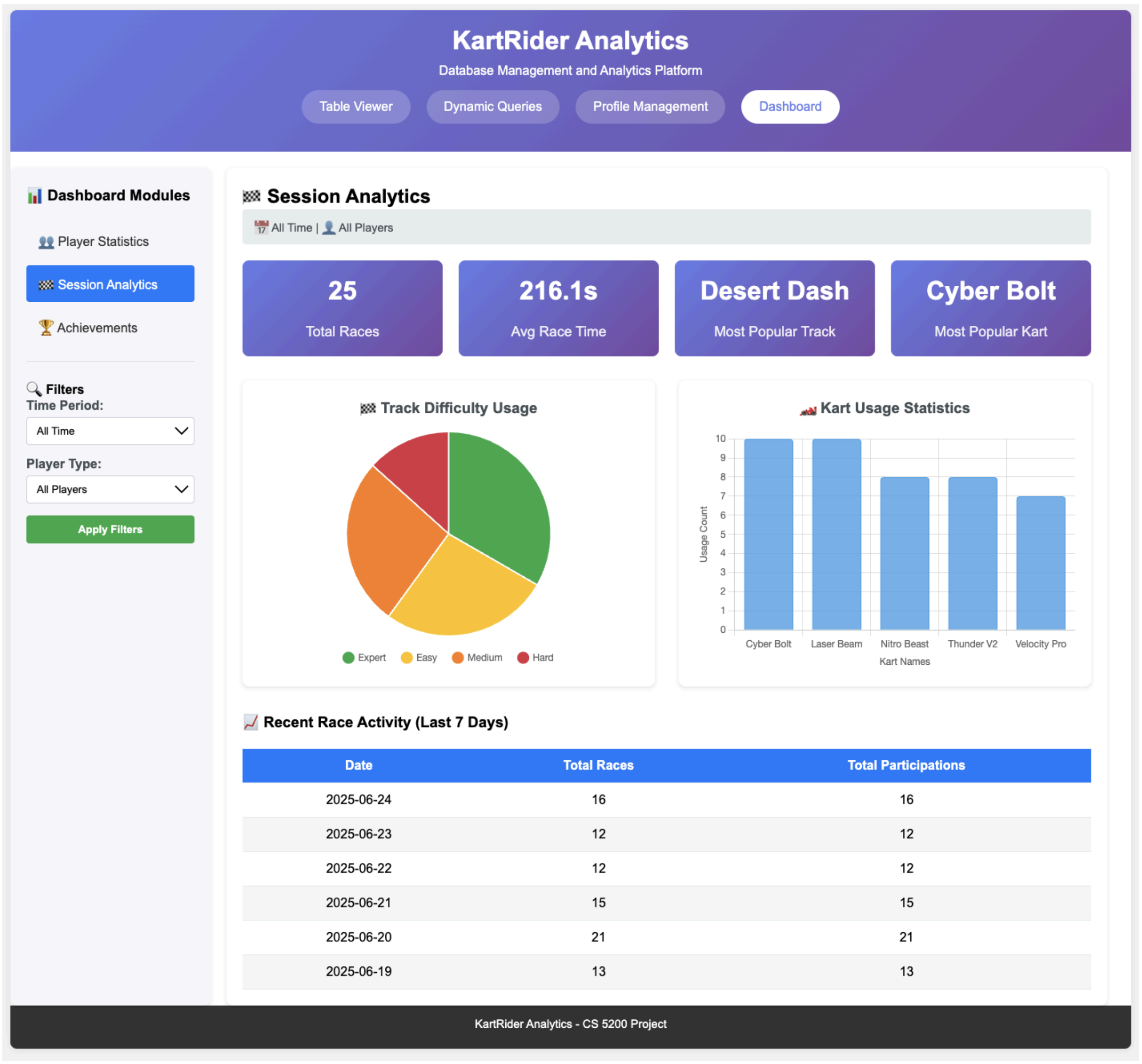
Chart Type: Pie Chart + Bar Chart + Summary Cards + Data Table

Data Source:

```sql
-- 1. Session Analytics Summary (KPI Cards)
SELECT
    (SELECT COUNT(*) FROM Race) AS TotalRaces,
    (SELECT ROUND(AVG(TotalTime), 1) FROM Participation WHERE
TotalTime IS NOT NULL) AS AvgRaceTime,
    (SELECT r.TrackName FROM Race r GROUP BY r.TrackName ORDER
BY COUNT(*) DESC LIMIT 1) AS MostPopularTrack,
    (SELECT k.KartName FROM Participation pt
     JOIN Kart k ON pt.KartID = k.KartID
     GROUP BY k.KartName ORDER BY COUNT(*) DESC LIMIT 1) AS
MostPopularKart;


-- 2. Most Popular Track
SELECT
    r.TrackName,
    COUNT(*) AS race_count
FROM Race r
GROUP BY r.TrackName
ORDER BY race_count DESC
LIMIT 1;

-- 3. Most Popular Kart
SELECT
    k.KartName,
    COUNT(pt.ParticipationID) AS usage_count
FROM Participation pt
    INNER JOIN Kart k ON pt.KartID = k.KartID
GROUP BY k.KartID, k.KartName
ORDER BY usage_count DESC
LIMIT 1;

-- 4. Track Difficulty Distribution (Pie Chart)
SELECT
    t.TrackDifficulty,
    COUNT(r.RaceID) AS RaceCount
FROM Race r
    INNER JOIN Track t ON r.TrackName = t.TrackName
GROUP BY t.TrackDifficulty
ORDER BY RaceCount DESC;
```

```
-- 5. Top 5 Kart Usage Statistics (Bar Chart)
SELECT
    k.KartName,
    CASE
        WHEN sk.KartID IS NOT NULL THEN 'Speed'
        WHEN ik.KartID IS NOT NULL THEN 'Item'
        ELSE 'Standard'
    END AS KartType,
    COUNT(pt.ParticipationID) AS UsageCount
FROM Participation pt
    INNER JOIN Kart k ON pt.KartID = k.KartID
    LEFT JOIN SpeedKart sk ON k.KartID = sk.KartID
    LEFT JOIN ItemKart ik ON k.KartID = ik.KartID
GROUP BY k.KartID, k.KartName, KartType
ORDER BY UsageCount DESC
LIMIT 5;

-- 6. Recent Race Activity (Last 7 Days Table)
SELECT
    DATE(r.RaceDate) AS Date,
    COUNT(r.RaceID) AS TotalRaces,
    COUNT(pt.ParticipationID) AS TotalParticipations
FROM Race r
    LEFT JOIN Participation pt ON r.RaceID = pt.RaceID
WHERE r.RaceDate >= DATE_SUB(NOW(), INTERVAL 7 DAY)
GROUP BY DATE(r.RaceDate)
ORDER BY Date DESC;

-- 7. Average Race Time Calculation
SELECT
    ROUND(AVG(TotalTime), 1) AS avg_race_time
FROM Participation
WHERE TotalTime IS NOT NULL;

-- 8. Daily Race Trends (Extended)
SELECT
    DATE(r.RaceDate) AS RaceDay,
    COUNT(*) AS DailyRaces,
    COUNT(DISTINCT pt.PlayerID) AS UniquePlayers,
    ROUND(AVG(pt.TotalTime), 1) AS AvgDailyTime
FROM Race r
    LEFT JOIN Participation pt ON r.RaceID = pt.RaceID
WHERE r.RaceDate >= DATE_SUB(NOW(), INTERVAL 15 DAY)
GROUP BY DATE(r.RaceDate)
```

```
ORDER BY RaceDay DESC
LIMIT 15;
```



Time Period: All Time

Player Type: All Players

Session Analytics

All Time | All Players

25	216.1s	Desert Dash	Cyber Bolt
Total Races	Avg Race Time	Most Popular Track	Most Popular Kart

Recent Race Activity (Last 7 Days)

Date	Total Races	Total Participations
2025-06-24	16	16
2025-06-23	12	12
2025-06-22	12	12
2025-06-21	15	15
2025-06-20	21	21
2025-06-19	13	13

2.3 Achievements Module

The Achievements Dashboard summarizes how players engage with in-game rewards. It displays total achievements, times earned, average per player, and completion rate. The dashboard highlights achievement popularity, distribution across players, the rarest achievement, and the top 5 most earned achievements. Filters allow analysis by time period and player type.

Key Features:

- Achievement Popularity Donut Chart
- Achievement Distribution Bar Chart
- Summary Cards: Total achievements, times earned, average per player, completion rate

- Rarest Achievement Highlight
- Top 5 Achievement Table: With description, points, earned count, and completion rate
- Filters: By time period and player type

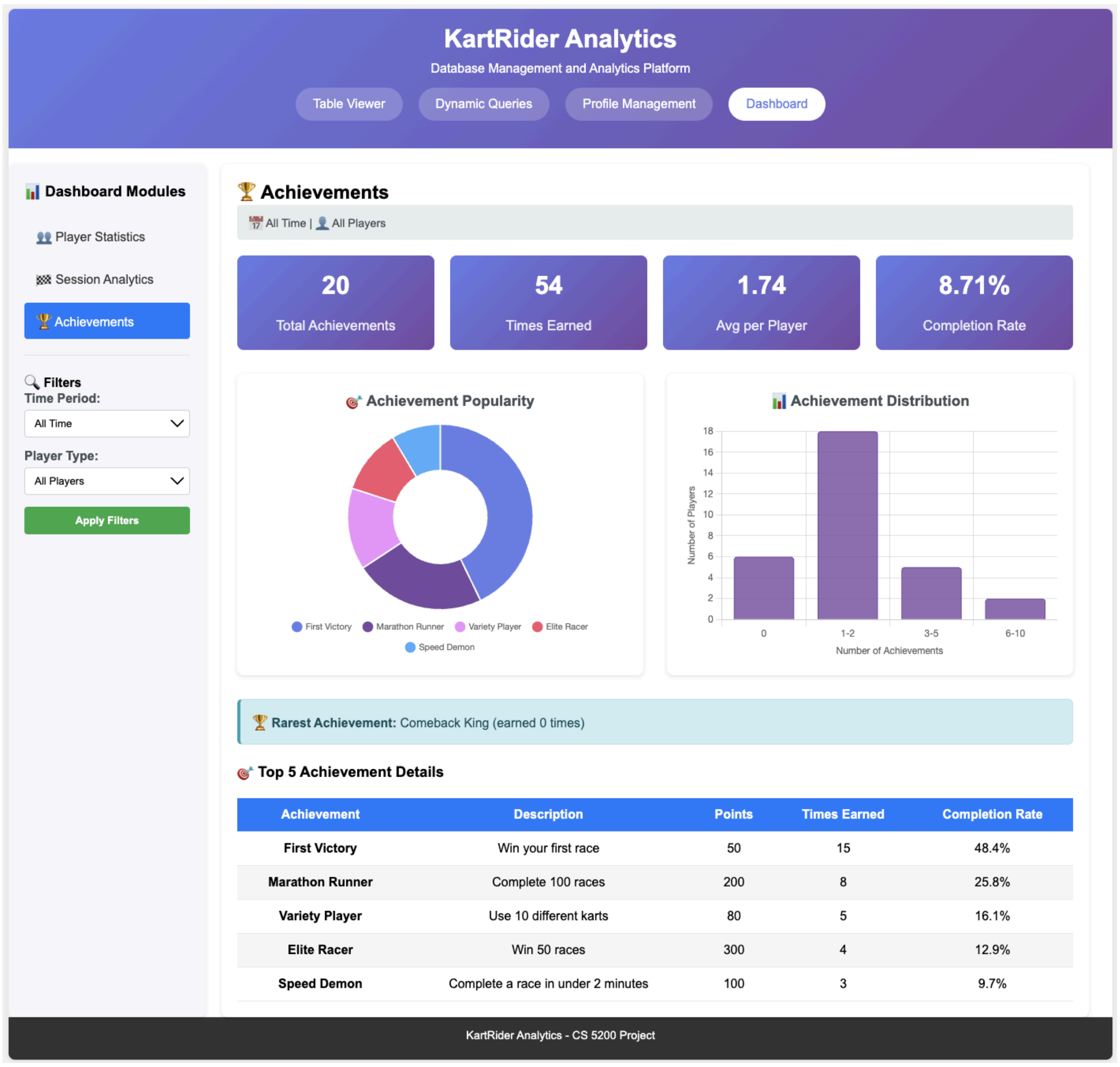
Chart Type: Donut Chart + Bar Chart + Summary Cards + Data Table

Data Source:

```
-- 1. Achievement Summary Statistics (KPI Cards)
SELECT
     (SELECT COUNT(*) FROM Achievement) AS TotalAchievements,
     (SELECT COUNT(*) FROM PlayerAchievement) AS TimesEarned,
     (SELECT ROUND(COUNT(*) * 1.0 / (SELECT COUNT(DISTINCT
PlayerID) FROM Player), 2)
      FROM PlayerAchievement) AS AvgPerPlayer,
     (SELECT ROUND(COUNT(*) * 100.0 /
                  ((SELECT COUNT(*) FROM Achievement) * (SELECT
COUNT(DISTINCT PlayerID) FROM Player)), 2)
      FROM PlayerAchievement) AS CompletionRate;

-- 2. Top 5 Achievement Details (Main Table)
SELECT
    a.AchievementName AS Achievement,
    a.Description,
    a.PointsAwarded AS Points,
    COUNT(pa.PlayerID) AS TimesEarned,
    ROUND((COUNT(pa.PlayerID) * 100.0 / (SELECT COUNT(DISTINCT
PlayerID) FROM Player)), 1) AS CompletionRate
FROM Achievement a
    LEFT JOIN PlayerAchievement pa ON a.AchievementID =
pa.AchievementID
GROUP BY a.AchievementID, a.AchievementName, a.Description,
a.PointsAwarded
ORDER BY TimesEarned DESC
LIMIT 5;

-- 3. Achievement Popularity (Pie Chart Data)
SELECT
    a.AchievementName,
    COUNT(pa.PlayerID) AS EarnedCount
FROM Achievement a
    LEFT JOIN PlayerAchievement pa ON a.AchievementID =
pa.AchievementID
GROUP BY a.AchievementID, a.AchievementName
HAVING EarnedCount > 0
```

```
ORDER BY EarnedCount DESC
LIMIT 5;

-- 4. Achievement Distribution (Bar Chart)
SELECT
    CASE
        WHEN achievement_count = 0 THEN '0 achievements'
        WHEN achievement_count BETWEEN 1 AND 2 THEN '1-2
achievements'
        WHEN achievement_count BETWEEN 3 AND 5 THEN '3-5
achievements'
        WHEN achievement_count BETWEEN 6 AND 10 THEN '6-10
achievements'
        ELSE '10+ achievements'
    END AS AchievementRange,
    COUNT(*) AS NumberOfPlayers
FROM (
    SELECT
        p.PlayerID,
        COUNT(pa.AchievementID) AS achievement_count
    FROM Player p
        LEFT JOIN PlayerAchievement pa ON p.PlayerID =
pa.PlayerID
    GROUP BY p.PlayerID
) player_achievement_counts
GROUP BY AchievementRange
ORDER BY
    CASE AchievementRange
        WHEN '0 achievements' THEN 1
        WHEN '1-2 achievements' THEN 2
        WHEN '3-5 achievements' THEN 3
        WHEN '6-10 achievements' THEN 4
        ELSE 5
    END;

-- 5. Rarest Achievement
SELECT
    a.AchievementName,
    COUNT(pa.PlayerID) AS EarnedCount
FROM Achievement a
    LEFT JOIN PlayerAchievement pa ON a.AchievementID =
pa.AchievementID
GROUP BY a.AchievementID, a.AchievementName
ORDER BY EarnedCount ASC, a.AchievementName
```

```
LIMIT 1;
```



KartRider Analytics

Database Management and Analytics Platform

Table Viewer | Dynamic Queries | Profile Management | Dashboard

Dashboard Modules

- Player Statistics
- Session Analytics
- Achievements

Filters

Time Period: All Time

Player Type: All Players

Apply Filters

Achievements

All Time | All Players

20	54	1.74	8.71%
Total Achievements	Times Earned	Avg per Player	Completion Rate

Rarest Achievement: Comeback King (earned 0 times)

Top 5 Achievement Details

Achievement	Description	Points	Times Earned	Completion Rate
First Victory	Win your first race	50	15	48.4%
Marathon Runner	Complete 100 races	200	8	25.8%
Variety Player	Use 10 different karts	80	5	16.1%
Elite Racer	Win 50 races	300	4	12.9%
Speed Demon	Complete a race in under 2 minutes	100	3	9.7%

KartRider Analytics - CS 5200 Project

3. Conclusion

The KartRider Analytics application successfully delivers a fully functional database-driven web platform with comprehensive CRUD operations, dynamic queries, and interactive dashboards. The project demonstrates strong database design principles, ensuring data integrity through normalization (3NF/BCNF) and cascade operations.

All core requirements are fully achieved, including six key SQL operations: Join, Aggregation, Nested Aggregation with Group-By, Filtering & Ranking, Update, and Delete with Cascade. Additional features such as a Register operation and a Custom Query interface further enhance the system’s functionality and flexibility.

The dashboards provide clear visual representations of player statistics, session data, and achievement progress, supported by interactive filters for time and player type. Indexing strategies are applied to maintain efficient query performance.

The analytics reveal meaningful gaming patterns: 100% player activity rate, balanced difficulty preferences, and clear performance hierarchies among 30+ active players. The system tracks 135+ race participation across 25 events, demonstrating scalability and real-world applicability.

In summary, this project meets the specified objectives while strengthening our understanding of relational databases, query optimization, and data visualization within a web application context.