SQL CODE DOCUMENTATION
Centennial Conference Track & Field Database Project

This document contains all SQL queries used throughout the project, organized by location and purpose.

1. DATABASE SCHEMA

FILE: db_generating/table_generation.sql
PURPOSE: Creates all database tables and defines schema structure

```
-- GeographicLocation Table
DROP TABLE IF EXISTS GeographicLocation CASCADE;
CREATE TABLE GeographicLocation (
    LocationID      SERIAL PRIMARY KEY,
    CityName        VARCHAR(100) NOT NULL,
    StateCode       VARCHAR(2) NOT NULL,
    CountryCode     VARCHAR(2) NOT NULL
);

-- School Table
DROP TABLE IF EXISTS School CASCADE;
CREATE TABLE School (
    SchoolID        VARCHAR(20) PRIMARY KEY,
    SchoolName      VARCHAR(100) NOT NULL,
    LocationID      INT NOT NULL REFERENCES GeographicLocation(LocationID),
    StreetAddress   VARCHAR(100) NOT NULL,
    EnrollmentSize  INT NOT NULL,
    HasIndoorFacility BOOLEAN NOT NULL
);

-- TrackMeet Table
DROP TABLE IF EXISTS TrackMeet CASCADE;
CREATE TABLE TrackMeet (
    MeetID          INT PRIMARY KEY,
    MeetName        VARCHAR(200) NOT NULL,
    StartDate       DATE NOT NULL,
    EndDate         DATE NOT NULL
);

-- Athlete Table
DROP TABLE IF EXISTS Athlete CASCADE;
CREATE TABLE Athlete (
    AthleteID       INT PRIMARY KEY,
    AthleteLastName VARCHAR(100) NOT NULL,
    AthleteFirstName VARCHAR(100) NOT NULL,
    Gender          VARCHAR(1) NOT NULL CHECK (Gender IN ('M', 'F'))
);
```

```
-- AthleteSeason Table
DROP TABLE IF EXISTS AthleteSeason CASCADE;
CREATE TABLE AthleteSeason (
   AthleteSeasonID SERIAL PRIMARY KEY,
   AthleteID       INT NOT NULL REFERENCES Athlete(AthleteID),
   SchoolID        VARCHAR(20) NOT NULL REFERENCES School(SchoolID),
   SeasonType      VARCHAR(10) NOT NULL CHECK (SeasonType IN ('Indoor', 'Outdoor')),
   SeasonYear      INT NOT NULL,
   ClassYear       VARCHAR(2) NOT NULL CHECK (ClassYear IN ('FR', 'SO', 'JR', 'SR')),
   UNIQUE (AthleteID, SeasonType, SeasonYear)
);

-- TrackEvent Table
DROP TABLE IF EXISTS TrackEvent CASCADE;
CREATE TABLE TrackEvent (
   EventID         INT PRIMARY KEY,
   EventName       VARCHAR(20) NOT NULL,
   EventType       VARCHAR(8) NOT NULL CHECK (EventType IN ('sprints', 'distance', 'jumps', 'throws',
'combined')),
   MeasureUnit     VARCHAR(7) NOT NULL CHECK (MeasureUnit IN ('seconds', 'meters', 'points')),
   IsRelay         BOOLEAN NOT NULL
);

-- RelayTeam Table
DROP TABLE IF EXISTS RelayTeam CASCADE;
CREATE TABLE RelayTeam (
   RelayTeamID     SERIAL PRIMARY KEY,
   SchoolID        VARCHAR(20) NOT NULL REFERENCES School(SchoolID),
   EventID         INT NOT NULL REFERENCES TrackEvent(EventID),
   MeetID          INT NOT NULL REFERENCES TrackMeet(MeetID)
);

-- RelayTeamMembers Table
DROP TABLE IF EXISTS RelayTeamMembers CASCADE;
CREATE TABLE RelayTeamMembers (
   RelayTeamID       INT NOT NULL REFERENCES RelayTeam(RelayTeamID),
   AthleteSeasonID   INT NOT NULL REFERENCES AthleteSeason(AthleteSeasonID),
   LegNum            INT NOT NULL CHECK (LegNum IN (1, 2, 3, 4)),
   PRIMARY KEY (RelayTeamID, AthleteSeasonID)
);

-- Performance Table
DROP TABLE IF EXISTS Performance CASCADE;
CREATE TABLE Performance (
   PerformanceID    SERIAL PRIMARY KEY,
   MeetID           INT NOT NULL REFERENCES TrackMeet(MeetID),
   EventID          INT NOT NULL REFERENCES TrackEvent(EventID),
   AthleteSeasonID  INT REFERENCES AthleteSeason(AthleteSeasonID),
```

```
    RelayTeamID      INT REFERENCES RelayTeam(RelayTeamID),
    ResultValue      DECIMAL(8, 2) NOT NULL,
    WindGauge        DECIMAL(3, 1),
    CHECK ((RelayTeamID IS NULL AND AthleteSeasonID IS NOT NULL) OR (AthleteSeasonID IS NULL
AND RelayTeamID IS NOT NULL))
);

-- CentennialConferenceEvents Table
DROP TABLE IF EXISTS CentennialConferenceEvents CASCADE;
CREATE TABLE CentennialConferenceEvents (
    EventID         INT PRIMARY KEY,
    Indoor          BOOLEAN NOT NULL,
    Outdoor         BOOLEAN NOT NULL
);
```

FILE: db_generating/add_manual_info.sql
PURPOSE: Inserts initial geographic and school data, and Centennial Conference events

```
-- Insert Geographic Locations
INSERT INTO GeographicLocation (LocationID, CityName, StateCode, CountryCode) VALUES
(1, 'Baltimore', 'MD', 'US'),
(2, 'Collegeville', 'PA', 'US'),
(3, 'Carlisle', 'PA', 'US'),
(4, 'Lancaster', 'PA', 'US'),
(5, 'Gettysburg', 'PA', 'US'),
(6, 'Haverford', 'PA', 'US'),
(7, 'Westminster', 'MD', 'US'),
(8, 'Allentown', 'PA', 'US'),
(9, 'Bryn Mawr', 'PA', 'US'),
(10, 'Swarthmore', 'PA', 'US');

-- Insert Schools
INSERT INTO School (SchoolID, SchoolName, LocationID, StreetAddress, EnrollmentSize,
HasIndoorFacility) VALUES
('Johns_Hopkins', 'Johns Hopkins University', 1, '3400 North Charles Street', 7000, False),
('Ursinus', 'Ursinus College', 2, '601 East Main Street', 1500, True),
('Dickinson_College', 'Dickinson College', 3, '28 North College Street', 2200, True),
('Franklin__Marshall', 'Franklin & Marshall College', 4, '415 Harrisburg Avenue', 2250, True),
('Gettysburg', 'Gettysburg College', 5, '300 North Washington Street', 2400, True),
('Haverford', 'Haverford College', 6, '370 Lancaster Avenue', 1450, True),
('McDaniel', 'McDaniel College', 7, '2 College Hill', 3000, True),
('Muhlenberg', 'Muhlenberg College', 8, '2400 Chew Street', 1950, True),
('Bryn_Mawr', 'Bryn Mawr College', 9, '101 North Merion Avenue', 1750, True),
('Swarthmore', 'Swarthmore College', 10, '500 College Avenue', 1700, True);

-- Insert Centennial Conference Events (Indoor and Outdoor championships)
INSERT INTO CentennialConferenceEvents (EventID, Indoor, Outdoor) VALUES
-- Indoor Championships Only
```

```
(69, TRUE, FALSE),  -- Weight Throw
(80, TRUE, FALSE),  -- Heptathlon (indoor)
(81, TRUE, FALSE),  -- Pentathlon
(46, TRUE, FALSE),  -- 60 Meters
(50, TRUE, FALSE),  -- 60 Hurdles
(49, TRUE, FALSE),  -- 1000 Meters
(57, TRUE, FALSE),  -- Mile
(60, TRUE, FALSE),  -- 3000 Meters
(62, TRUE, FALSE),  -- 5000 Meters
(51, TRUE, FALSE),  -- 200 Meters
(53, TRUE, FALSE),  -- 400 Meters
(54, TRUE, FALSE),  -- 800 Meters
(64, TRUE, FALSE),  -- High Jump
(65, TRUE, FALSE),  -- Pole Vault
(66, TRUE, FALSE),  -- Long Jump
(67, TRUE, FALSE),  -- Triple Jump
(68, TRUE, FALSE),  -- Shot Put
(71, TRUE, FALSE),  -- 4 x 200 Relay
(73, TRUE, FALSE),  -- 4 x 400 Relay
(74, TRUE, FALSE),  -- 4 x 800 Relay
(78, TRUE, FALSE),  -- Distance Medley Relay
-- Outdoor Championships Only
(39, FALSE, TRUE),  -- Decathlon
(40, FALSE, TRUE),  -- Heptathlon (outdoor)
(6, FALSE, TRUE),   -- 100 Meters
(7, FALSE, TRUE),   -- 200 Meters
(11, FALSE, TRUE),  -- 400 Meters
(12, FALSE, TRUE),  -- 800 Meters
(13, FALSE, TRUE),  -- 1500 Meters
(21, FALSE, TRUE),  -- 5000 Meters
(22, FALSE, TRUE),  -- 10,000 Meters
(4, FALSE, TRUE),   -- 100 Hurdles
(5, FALSE, TRUE),   -- 110 Hurdles
(9, FALSE, TRUE),   -- 400 Hurdles
(19, FALSE, TRUE),  -- 3000 Steeplechase
(23, FALSE, TRUE),  -- High Jump
(24, FALSE, TRUE),  -- Pole Vault
(25, FALSE, TRUE),  -- Long Jump
(26, FALSE, TRUE),  -- Triple Jump
(30, FALSE, TRUE),  -- Shot Put
(27, FALSE, TRUE),  -- Discus
(28, FALSE, TRUE),  -- Hammer
(29, FALSE, TRUE),  -- Javelin
(31, FALSE, TRUE),  -- 4 x 100 Relay
(33, FALSE, TRUE),  -- 4 x 400 Relay
(34, FALSE, TRUE);  -- 4 x 800 Relay
```

FILE: db_generating/analysis_tables.sql

PURPOSE: Creates analysis table for linear regression predictions

```
DROP TABLE IF EXISTS LinearRegressionPredictions CASCADE;
CREATE TABLE LinearRegressionPredictions (
   EventID         INT NOT NULL REFERENCES TrackEvent(EventID),
   EventName       VARCHAR(20) NOT NULL,
   EventType       VARCHAR(8) NOT NULL CHECK (EventType IN ('sprints', 'distance', 'jumps', 'throws',
'combined')),
   Gender          VARCHAR(1) NOT NULL CHECK (Gender IN ('M', 'F', 'X')),
   SchoolID        VARCHAR(20) NOT NULL REFERENCES School(SchoolID),
   AthleteID       INT NOT NULL,
   AthleteFirstName VARCHAR(100) NOT NULL,
   AthleteLastName  VARCHAR(100) NOT NULL,
   predictedresult  DECIMAL(8, 2) NOT NULL,
   seasonType      VARCHAR(8) NOT NULL CHECK (seasonType IN ('Outdoor', 'Indoor')),
   seasonYear      INT NOT NULL
);
```

2. API ROUTES - ATHLETES

FILE: predict-the-centenni-podium/app/api/athletes/[id]/route.ts
PURPOSE: Get detailed information about a specific athlete

QUERY 1: Get Athlete Basic Info
PURPOSE: Retrieves basic athlete information (ID, name, gender)
SQL:

```
SELECT
   a.AthleteID,
   a.AthleteFirstName,
   a.AthleteLastName,
   a.Gender
FROM Athlete a
WHERE a.AthleteID = $1
```

QUERY 2: Get Athlete Seasons
PURPOSE: Gets all seasons an athlete has competed in, with school information
SQL:

```
SELECT
   ats.AthleteSeasonID,
   ats.SeasonType,
   ats.SeasonYear,
   ats.ClassYear,
   s.SchoolName,
   s.SchoolID
FROM AthleteSeason ats
JOIN School s ON ats.SchoolID = s.SchoolID
WHERE ats.AthleteID = $1
```

```
ORDER BY ats.SeasonYear DESC, ats.SeasonType

QUERY 3: Get Personal Bests (Individual Events)
PURPOSE: Gets all-time best performance per event, separated by Indoor/Outdoor
SQL:
SELECT DISTINCT ON (e.EventID, ats.SeasonType)
    e.EventID,
    e.EventName,
    e.EventType,
    e.MeasureUnit,
    ats.SeasonType,
    p.ResultValue AS PersonalBest,
    tm.MeetName AS PBMeet,
    tm.StartDate AS PBDate
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE ats.AthleteID = $1
    AND e.IsRelay = FALSE
    AND p.ResultValue IS NOT NULL
ORDER BY e.EventID, ats.SeasonType,
    CASE WHEN e.EventType IN ('throws', 'jumps', 'combined') THEN -p.ResultValue ELSE p.ResultValue
END ASC,
    tm.StartDate DESC

QUERY 4: Get Season Bests (Individual Events)
PURPOSE: Gets best performance per event per season
SQL:
SELECT DISTINCT ON (ats.SeasonYear, ats.SeasonType, e.EventID)
    ats.SeasonYear,
    ats.SeasonType,
    e.EventName,
    p.ResultValue AS SeasonBest
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE ats.AthleteID = $1
    AND e.IsRelay = FALSE
    AND p.ResultValue IS NOT NULL
ORDER BY ats.SeasonYear DESC, ats.SeasonType, e.EventID,
    CASE WHEN e.EventType IN ('throws', 'jumps', 'combined') THEN -p.ResultValue ELSE p.ResultValue
END ASC

QUERY 5: Get Performance History (Individual Events)
PURPOSE: Gets all individual event performances in chronological order
SQL:
SELECT
    p.PerformanceID,
```

```
    e.EventName,
    e.EventType,
    p.ResultValue,
    p.WindGauge,
    tm.MeetName,
    tm.StartDate,
    ats.SeasonYear,
    ats.SeasonType
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE ats.AthleteID = $1
    AND e.IsRelay = FALSE
ORDER BY tm.StartDate DESC
```

QUERY 6: Get Trend Data (Individual Events)
PURPOSE: Gets all performance data for trend analysis, grouped by event and season type
SQL:

```
SELECT
    e.EventName,
    ats.SeasonType,
    p.ResultValue,
    tm.StartDate
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE ats.AthleteID = $1
    AND e.IsRelay = FALSE
    AND p.ResultValue IS NOT NULL
ORDER BY e.EventName, ats.SeasonType, tm.StartDate ASC
```

QUERY 7: Get Relay Personal Bests
PURPOSE: Gets all-time best relay performance per event, separated by Indoor/Outdoor
SQL:

```
WITH UniqueRelayPerformances AS (
    SELECT DISTINCT ON (p.PerformanceID)
        p.PerformanceID,
        e.EventID,
        e.EventName,
        e.EventType,
        e.MeasureUnit,
        ats.SeasonType,
        p.ResultValue AS PersonalBest,
        tm.MeetName AS PBMeet,
        tm.StartDate AS PBDate,
        s.SchoolName AS SchoolName
    FROM Performance p
```

```
    JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
    JOIN RelayTeamMembers rtm ON rt.RelayTeamID = rtm.RelayTeamID
    JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
    JOIN TrackEvent e ON p.EventID = e.EventID
    JOIN TrackMeet tm ON p.MeetID = tm.MeetID
    JOIN School s ON rt.SchoolID = s.SchoolID
    WHERE ats.AthleteID = $1
        AND e.IsRelay = TRUE
        AND p.ResultValue IS NOT NULL
    ORDER BY p.PerformanceID
)
SELECT DISTINCT ON (EventID, SeasonType)
    EventID,
    EventName,
    EventType,
    MeasureUnit,
    SeasonType,
    PersonalBest,
    PBMeet,
    PBDate,
    SchoolName
FROM UniqueRelayPerformances
ORDER BY EventID, SeasonType, PersonalBest ASC, PBDate DESC

QUERY 8: Get Relay Season Bests
PURPOSE: Gets best relay performance per event per season
SQL:
WITH UniqueRelayPerformances AS (
    SELECT DISTINCT ON (p.PerformanceID)
        p.PerformanceID,
        ats.SeasonYear,
        ats.SeasonType,
        e.EventID,
        e.EventName,
        p.ResultValue AS SeasonBest,
        s.SchoolName AS SchoolName
    FROM Performance p
    JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
    JOIN RelayTeamMembers rtm ON rt.RelayTeamID = rtm.RelayTeamID
    JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
    JOIN TrackEvent e ON p.EventID = e.EventID
    JOIN School s ON rt.SchoolID = s.SchoolID
    WHERE ats.AthleteID = $1
        AND e.IsRelay = TRUE
        AND p.ResultValue IS NOT NULL
    ORDER BY p.PerformanceID
)
SELECT DISTINCT ON (SeasonYear, SeasonType, EventID)
    SeasonYear,
```

```
    SeasonType,
    EventName,
    SeasonBest,
    SchoolName
FROM UniqueRelayPerformances
ORDER BY SeasonYear DESC, SeasonType, EventID, SeasonBest ASC
```

QUERY 9: Get Relay Performance History
PURPOSE: Gets all relay performances in chronological order
SQL:

```
SELECT DISTINCT ON (p.PerformanceID)
    p.PerformanceID,
    e.EventName,
    e.EventType,
    p.ResultValue,
    tm.MeetName,
    tm.StartDate,
    ats.SeasonYear,
    ats.SeasonType,
    s.SchoolName AS SchoolName
FROM Performance p
JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
JOIN RelayTeamMembers rtm ON rt.RelayTeamID = rtm.RelayTeamID
JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
JOIN School s ON rt.SchoolID = s.SchoolID
WHERE ats.AthleteID = $1
    AND e.IsRelay = TRUE
ORDER BY p.PerformanceID, tm.StartDate DESC
```

QUERY 10: Get Relay Trend Data
PURPOSE: Gets all relay performance data for trend analysis
SQL:

```
SELECT DISTINCT ON (p.PerformanceID)
    e.EventName,
    ats.SeasonType,
    p.ResultValue,
    tm.StartDate
FROM Performance p
JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
JOIN RelayTeamMembers rtm ON rt.RelayTeamID = rtm.RelayTeamID
JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE ats.AthleteID = $1
    AND e.IsRelay = TRUE
    AND p.ResultValue IS NOT NULL
ORDER BY p.PerformanceID, e.EventName, ats.SeasonType, tm.StartDate ASC
```

FILE: predict-the-centenni-podium/app/api/athletes/route.ts
PURPOSE: Search and list athletes

QUERY 1: Search Athletes (Dynamic Query)
PURPOSE: Searches for athletes by name and optionally filters by school
SQL BASE:

```
SELECT DISTINCT
   a.AthleteID,
   a.AthleteFirstName,
   a.AthleteLastName,
   a.Gender,
   s.SchoolName,
   s.SchoolID
FROM Athlete a
JOIN AthleteSeason ats ON a.AthleteID = ats.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID

-- If search term has 2+ words, adds:
WHERE (
   (a.AthleteFirstName ILIKE $1 AND a.AthleteLastName ILIKE $2)
   OR a.AthleteLastName ILIKE $3
   OR a.AthleteFirstName ILIKE $3
)

-- If single word search, adds:
WHERE (a.AthleteFirstName ILIKE $1 OR a.AthleteLastName ILIKE $1)

-- If school filter, adds:
AND ats.SchoolID = $N

ORDER BY a.AthleteLastName, a.AthleteFirstName LIMIT 100
```

==========

3. API ROUTES - SCHOOLS

==========

FILE: predict-the-centenni-podium/app/api/schools/route.ts
PURPOSE: Get list of all schools

QUERY 1: Get All Schools
PURPOSE: Retrieves all schools with location information
SQL:

```
SELECT
   s.SchoolID,
   s.SchoolName,
   s.EnrollmentSize,
```

```
    s.HasIndoorFacility,
    g.CityName,
    g.StateCode
FROM School s
JOIN GeographicLocation g ON s.LocationID = g.LocationID
ORDER BY s.SchoolName
```

FILE: predict-the-centenni-podium/app/api/schools/[id]/route.ts
PURPOSE: Get detailed information about a specific school

QUERY 1: Get School Info
PURPOSE: Gets basic school information including location
SQL:

```
SELECT
    s.SchoolID,
    s.SchoolName,
    s.EnrollmentSize,
    s.HasIndoorFacility,
    s.StreetAddress,
    g.CityName,
    g.StateCode
FROM School s
JOIN GeographicLocation g ON s.LocationID = g.LocationID
WHERE s.SchoolID = $1
```

QUERY 2: Get School Roster
PURPOSE: Gets roster for a specific season, filtered by gender
SQL:

```
SELECT
    a.AthleteID,
    a.AthleteFirstName,
    a.AthleteLastName,
    a.Gender,
    ats.ClassYear
FROM Athlete a
JOIN AthleteSeason ats ON a.AthleteID = ats.AthleteID
WHERE ats.SchoolID = $1
    AND ats.SeasonYear = $2
    AND ats.SeasonType = $3
    AND a.Gender = $4
ORDER BY a.AthleteLastName, a.AthleteFirstName
```

QUERY 3: Get School Records (Individual Events) - Top 5
PURPOSE: Gets all-time top 5 performances per event for the school
SQL:

```
WITH RankedPerformances AS (
    SELECT
        e.EventID,
```

```
        e.EventName,
        e.EventType,
        e.MeasureUnit,
        p.ResultValue AS SchoolRecord,
        a.AthleteID,
        a.AthleteFirstName,
        a.AthleteLastName,
        ats.SeasonYear,
        ROW_NUMBER() OVER (
            PARTITION BY e.EventID
            ORDER BY
                CASE WHEN e.EventType IN ('throws', 'jumps', 'combined') THEN -p.ResultValue ELSE
p.ResultValue END ASC
        ) AS rank
    FROM Performance p
    JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
    JOIN Athlete a ON ats.AthleteID = a.AthleteID
    JOIN TrackEvent e ON p.EventID = e.EventID
    WHERE ats.SchoolID = $1
        AND e.IsRelay = FALSE
        AND p.ResultValue IS NOT NULL
        AND a.Gender = $2
        AND ats.SeasonType = $3
)
SELECT
    EventID,
    EventName,
    EventType,
    MeasureUnit,
    SchoolRecord,
    AthleteID,
    AthleteFirstName,
    AthleteLastName,
    SeasonYear
FROM RankedPerformances
WHERE rank <= 5
ORDER BY EventType, EventName, rank

QUERY 4: Get Season Bests (Individual Events) - Top 5
PURPOSE: Gets top 5 performances per event for current season
SQL:
WITH RankedSeasonPerformances AS (
    SELECT
        e.EventID,
        e.EventName,
        e.EventType,
        p.ResultValue AS SeasonBest,
        a.AthleteID,
        a.AthleteFirstName,
```

```
        a.AthleteLastName,
        m.MeetName,
        m.StartDate,
        ROW_NUMBER() OVER (
            PARTITION BY e.EventID
            ORDER BY
                CASE WHEN e.EventType IN ('throws', 'jumps', 'combined') THEN -p.ResultValue ELSE
p.ResultValue END ASC
        ) AS rank
    FROM Performance p
    JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
    JOIN Athlete a ON ats.AthleteID = a.AthleteID
    JOIN TrackEvent e ON p.EventID = e.EventID
    JOIN TrackMeet m ON p.MeetID = m.MeetID
    WHERE ats.SchoolID = $1
        AND ats.SeasonYear = $2
        AND ats.SeasonType = $3
        AND e.IsRelay = FALSE
        AND p.ResultValue IS NOT NULL
        AND a.Gender = $4
        AND m.StartDate >= $5::date
        AND m.StartDate <= $6::date
)
SELECT
    EventID,
    EventName,
    EventType,
    SeasonBest,
    AthleteID,
    AthleteFirstName,
    AthleteLastName,
    MeetName,
    StartDate
FROM RankedSeasonPerformances
WHERE rank <= 5
ORDER BY EventType, EventName, rank

QUERY 5: Get Class Breakdown
PURPOSE: Counts athletes by class year (FR, SO, JR, SR) for the season
SQL:
SELECT
    ats.ClassYear,
    COUNT(DISTINCT a.AthleteID) AS Count
FROM AthleteSeason ats
JOIN Athlete a ON ats.AthleteID = a.AthleteID
WHERE ats.SchoolID = $1
    AND ats.SeasonYear = $2
    AND ats.SeasonType = $3
    AND a.Gender = $4
```

```
GROUP BY ats.ClassYear
ORDER BY ats.ClassYear
```

QUERY 6: Get Relay Records - Top 5
PURPOSE: Gets all-time top 5 relay performances per event
SQL:

```
WITH UniqueRelayPerformances AS (
    SELECT DISTINCT ON (p.PerformanceID)
        p.PerformanceID,
        e.EventID,
        e.EventName,
        e.EventType,
        e.MeasureUnit,
        p.ResultValue AS SchoolRecord,
        ats.SeasonYear
    FROM Performance p
    JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
    JOIN RelayTeamMembers rtm ON rt.RelayTeamID = rtm.RelayTeamID
    JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
    JOIN Athlete a ON ats.AthleteID = a.AthleteID
    JOIN TrackEvent e ON p.EventID = e.EventID
    WHERE rt.SchoolID = $1
        AND e.IsRelay = TRUE
        AND p.ResultValue IS NOT NULL
        AND a.Gender = $2
        AND ats.SeasonType = $3
    ORDER BY p.PerformanceID
),
RankedRelayPerformances AS (
    SELECT
        EventID,
        EventName,
        EventType,
        MeasureUnit,
        SchoolRecord,
        SeasonYear,
        ROW_NUMBER() OVER (
            PARTITION BY EventID
            ORDER BY SchoolRecord ASC
        ) AS rank
    FROM UniqueRelayPerformances
)
SELECT
    EventID,
    EventName,
    EventType,
    MeasureUnit,
    SchoolRecord,
    SeasonYear
```

```
FROM RankedRelayPerformances
WHERE rank <= 5
ORDER BY EventType, EventName, rank

QUERY 7: Get Relay Season Bests - Top 5
PURPOSE: Gets top 5 relay performances per event for current season
SQL:
WITH UniqueRelaySeasonPerformances AS (
    SELECT DISTINCT ON (p.PerformanceID)
        p.PerformanceID,
        e.EventID,
        e.EventName,
        e.EventType,
        p.ResultValue AS SeasonBest,
        m.MeetName,
        m.StartDate
    FROM Performance p
    JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
    JOIN RelayTeamMembers rtm ON rt.RelayTeamID = rtm.RelayTeamID
    JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
    JOIN Athlete a ON ats.AthleteID = a.AthleteID
    JOIN TrackEvent e ON p.EventID = e.EventID
    JOIN TrackMeet m ON p.MeetID = m.MeetID
    WHERE rt.SchoolID = $1
        AND ats.SeasonYear = $2
        AND ats.SeasonType = $3
        AND e.IsRelay = TRUE
        AND p.ResultValue IS NOT NULL
        AND a.Gender = $4
        AND m.StartDate >= $5::date
        AND m.StartDate <= $6::date
    ORDER BY p.PerformanceID
),
RankedRelaySeasonPerformances AS (
    SELECT
        EventID,
        EventName,
        EventType,
        SeasonBest,
        MeetName,
        StartDate,
        ROW_NUMBER() OVER (
            PARTITION BY EventID
            ORDER BY SeasonBest ASC
        ) AS rank
    FROM UniqueRelaySeasonPerformances
)
SELECT
    EventID,
```

```
    EventName,
    EventType,
    SeasonBest,
    MeetName,
    StartDate
FROM RankedRelaySeasonPerformances
WHERE rank <= 5
ORDER BY EventType, EventName, rank
```

4. API ROUTES - MEETS

FILE: predict-the-centenni-podium/app/api/meets/route.ts
PURPOSE: Get list of meets with optional filtering

QUERY 1: Get Meets (Dynamic Query)
PURPOSE: Lists meets with performance and athlete counts, with optional filtering
SQL BASE:

```
SELECT
    tm.MeetID,
    tm.MeetName,
    tm.StartDate,
    tm.EndDate,
    COUNT(DISTINCT p.PerformanceID) AS PerformanceCount,
    COUNT(DISTINCT ats.AthleteID) AS AthleteCount
FROM TrackMeet tm
LEFT JOIN Performance p ON tm.MeetID = p.MeetID
LEFT JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID

-- If year and seasonType provided:
-- Indoor: WHERE tm.StartDate >= (year-1)-12-01 AND tm.StartDate <= year-03-15
-- Outdoor: WHERE tm.StartDate >= year-03-15 AND tm.StartDate <= year-06-30

-- If just year: WHERE EXTRACT(YEAR FROM tm.StartDate) = year

-- If search: AND tm.MeetName ILIKE '%search%'

GROUP BY tm.MeetID, tm.MeetName, tm.StartDate, tm.EndDate
ORDER BY tm.StartDate DESC LIMIT 100
```

FILE: predict-the-centenni-podium/app/api/meets/[id]/route.ts
PURPOSE: Get detailed information about a specific meet

QUERY 1: Get Meet Info
PURPOSE: Gets basic meet information
SQL:

```
SELECT
    tm.MeetID,
```

```
    tm.MeetName,
    tm.StartDate,
    tm.EndDate
FROM TrackMeet tm
WHERE tm.MeetID = $1

QUERY 2: Get Individual Results
PURPOSE: Gets all individual event results with rankings
SQL:
SELECT
    e.EventID,
    e.EventName,
    e.EventType,
    e.MeasureUnit,
    a.AthleteID,
    a.AthleteFirstName,
    a.AthleteLastName,
    a.Gender,
    s.SchoolName,
    p.ResultValue,
    p.WindGauge,
    ROW_NUMBER() OVER (
        PARTITION BY e.EventID, a.Gender
        ORDER BY
            CASE WHEN e.MeasureUnit = 'seconds' THEN p.ResultValue END ASC,
            CASE WHEN e.MeasureUnit != 'seconds' THEN p.ResultValue END DESC
    ) AS Place
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE p.MeetID = $1
    AND p.AthleteSeasonID IS NOT NULL
ORDER BY a.Gender, e.EventType, e.EventName, Place

QUERY 3: Get Relay Results
PURPOSE: Gets all relay results with rankings
SQL:
SELECT DISTINCT ON (p.PerformanceID)
    e.EventID,
    e.EventName,
    s.SchoolName,
    p.ResultValue,
    a.Gender,
    ROW_NUMBER() OVER (
        PARTITION BY e.EventID, a.Gender
        ORDER BY p.ResultValue ASC
    ) AS Place
```

```
FROM Performance p
JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
JOIN School s ON rt.SchoolID = s.SchoolID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN RelayTeamMembers rtm ON rt.RelayTeamID = rtm.RelayTeamID
JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
WHERE p.MeetID = $1
  AND p.RelayTeamID IS NOT NULL
ORDER BY p.PerformanceID, a.Gender, e.EventName, Place
```

QUERY 4: Get Team Scores
PURPOSE: Calculates team scores using 10-8-6-5-4-3-2-1 scoring system
SQL:

```
WITH RankedPerformances AS (
  SELECT
    COALESCE(ats.SchoolID, rt.SchoolID) AS SchoolID,
    a.Gender,
    e.EventID,
    e.MeasureUnit,
    ROW_NUMBER() OVER (
      PARTITION BY e.EventID, a.Gender
      ORDER BY
        CASE WHEN e.MeasureUnit = 'seconds' THEN p.ResultValue END ASC,
        CASE WHEN e.MeasureUnit != 'seconds' THEN p.ResultValue END DESC
    ) AS Place
  FROM Performance p
  JOIN TrackEvent e ON p.EventID = e.EventID
  LEFT JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
  LEFT JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
  LEFT JOIN Athlete a ON ats.AthleteID = a.AthleteID
  WHERE p.MeetID = $1
)
SELECT
  s.SchoolName,
  rp.Gender,
  SUM(
    CASE Place
      WHEN 1 THEN 10
      WHEN 2 THEN 8
      WHEN 3 THEN 6
      WHEN 4 THEN 5
      WHEN 5 THEN 4
      WHEN 6 THEN 3
      WHEN 7 THEN 2
      WHEN 8 THEN 1
      ELSE 0
    END
  ) AS TotalPoints
```

```
FROM RankedPerformances rp
JOIN School s ON rp.SchoolID = s.SchoolID
WHERE rp.Place <= 8
GROUP BY s.SchoolID, s.SchoolName, rp.Gender
ORDER BY rp.Gender, TotalPoints DESC
```

5. API ROUTES - STATISTICS

FILE: predict-the-centenni-podium/app/api/stats/route.ts
PURPOSE: Get database-wide statistics

QUERY 1: Count Schools
PURPOSE: Counts total number of schools
SQL:
```
SELECT COUNT(*) as count FROM School
```

QUERY 2: Count Athletes
PURPOSE: Counts total number of unique athletes
SQL:
```
SELECT COUNT(DISTINCT AthleteID) as count FROM Athlete
```

QUERY 3: Count Performances
PURPOSE: Counts total number of performances
SQL:
```
SELECT COUNT(*) as count FROM Performance
```

QUERY 4: Count Events
PURPOSE: Counts total number of unique events
SQL:
```
SELECT COUNT(DISTINCT EventID) as count FROM TrackEvent
```

QUERY 5: Count Meets
PURPOSE: Counts total number of unique meets
SQL:
```
SELECT COUNT(DISTINCT MeetID) as count FROM TrackMeet
```

QUERY 6: Get Season Range
PURPOSE: Gets the minimum and maximum season years in the database
SQL:
```
SELECT
  MIN(SeasonYear) as min_year,
  MAX(SeasonYear) as max_year
FROM AthleteSeason
```

FILE: predict-the-centenni-podium/app/api/randomStat/route.ts
PURPOSE: Returns random statistics queries for display

QUERY 1: All Schools

PURPOSE: Lists all school names
SQL:
```
SELECT schoolname FROM school
```

QUERY 2: Count People Named Mike
PURPOSE: Counts athletes with first name "Mike"
SQL:
```
SELECT COUNT(*) FROM athlete WHERE athletefirstname = 'Mike'
```

QUERY 3: Spencer Ye's Best 400m Results
PURPOSE: Gets personal best in 400m events for Spencer Ye
SQL:
```
SELECT MIN(p.ResultValue) AS PersonalBest
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE a.AthleteLastName = 'Ye'
   AND a.AthleteFirstName = 'Spencer'
   AND e.EventName LIKE '%400%'
   AND e.IsRelay = FALSE
GROUP BY a.AthleteFirstName, a.AthleteLastName, e.EventName
```

QUERY 4: Johns Hopkins 2025 Indoor Roster
PURPOSE: Gets full roster for Johns Hopkins 2025 Indoor season
SQL:
```
SELECT DISTINCT
   a.AthleteFirstName || ' ' || a.AthleteLastName || ' (' || ats.ClassYear || ')' AS FullNameInfo
FROM Athlete a
JOIN AthleteSeason ats ON a.AthleteID = ats.AthleteID
WHERE ats.SchoolID = 'Johns_Hopkins'
   AND ats.SeasonYear = 2025
   AND ats.SeasonType = 'Indoor'
```

QUERY 5: Mirra Klimov's Best Results Each Season
PURPOSE: Gets season bests for Mirra Klimov across all seasons
SQL:
```
SELECT
   ats.SeasonYear || ' ' || ats.SeasonType || ' ' || e.EventName || ' ' || MIN(p.ResultValue) AS SeasonInfo
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE a.AthleteLastName = 'Klimov'
   AND a.AthleteFirstName = 'Mirra'
   AND e.IsRelay = FALSE
GROUP BY ats.SeasonYear, ats.SeasonType, e.EventName
```

QUERY 6: Athletes with Most Relay Appearances

PURPOSE: Finds athletes who have participated in the most relays
SQL:
```
SELECT
    a.AthleteFirstName || ' ' || a.AthleteLastName || ' (' || s.SchoolName || ') ' || ' ' || COUNT(DISTINCT
rtm.RelayTeamID) AS FullNameInfo,
    COUNT(DISTINCT rtm.RelayTeamID) AS RelayCount
FROM RelayTeamMembers rtm
JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
GROUP BY a.AthleteID, a.AthleteFirstName, a.AthleteLastName, s.SchoolName
ORDER BY RelayCount DESC
LIMIT 10
```

QUERY 7: Number of Seniors by School (2025)
PURPOSE: Counts seniors from each school graduating in 2025
SQL:
```
SELECT
    s.SchoolName || ' ' || COUNT(DISTINCT a.AthleteID) AS Result,
    COUNT(DISTINCT a.AthleteID) AS SeniorCount
FROM AthleteSeason ats
JOIN School s ON ats.SchoolID = s.SchoolID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
WHERE ats.ClassYear = 'SR'
    AND ats.SeasonYear = 2025
GROUP BY s.SchoolID, s.SchoolName
ORDER BY SeniorCount DESC
```

6. API ROUTES - PREDICTIONS

FILE: predict-the-centenni-podium/app/api/predictions/route.ts
PURPOSE: Generate predictions for conference championships

QUERY 1: Get All Schools
PURPOSE: Gets list of all school IDs for team scoring
SQL:
```
SELECT schoolid FROM school
```

QUERY 2: Season Best Predictions - Track Events (Sprints/Distance)
PURPOSE: Gets season best performances for track events (running events)
SQL:
```
SELECT
    P.EventID,
    TE.EventName,
    TE.Eventtype,
    Ath.gender,
    P.AthleteSeasonID,
    A.schoolid,
    Ath.athletefirstname,
```

```
    Ath.athletelastname,
    Ath.athleteid,
    MIN(P.Resultvalue) AS predictedresult
FROM Performance AS P
JOIN CentennialConferenceEvents AS CCE ON P.EventID = CCE.EventID
JOIN AthleteSeason AS A ON P.AthleteSeasonID = A.AthleteSeasonID
JOIN Athlete AS Ath ON A.AthleteID = Ath.AthleteID
JOIN TrackEvent AS TE ON P.EventID = TE.EventID
WHERE Ath.Gender = '{gender}'
    AND A.SeasonYear = '{seasonYear}'
    AND A.SeasonType = '{seasonType}'
    AND (TE.Eventtype = 'sprints' OR TE.Eventtype = 'distance')
GROUP BY P.EventID, TE.EventName, TE.Eventtype, Ath.gender, P.AthleteSeasonID, A.schoolid,
Ath.athletefirstname, Ath.athletelastname, Ath.athleteid
```

QUERY 3: Season Best Predictions - Field Events (Throws/Jumps/Combined)
PURPOSE: Gets season best performances for field events
SQL:

```
SELECT
    P.EventID,
    TE.EventName,
    TE.Eventtype,
    Ath.gender,
    P.AthleteSeasonID,
    A.schoolid,
    Ath.athletefirstname,
    Ath.athletelastname,
    Ath.athleteid,
    MAX(P.Resultvalue) AS predictedresult
FROM Performance AS P
JOIN CentennialConferenceEvents AS CCE ON P.EventID = CCE.EventID
JOIN AthleteSeason AS A ON P.AthleteSeasonID = A.AthleteSeasonID
JOIN Athlete AS Ath ON A.AthleteID = Ath.AthleteID
JOIN TrackEvent AS TE ON P.EventID = TE.EventID
WHERE Ath.Gender = '{gender}'
    AND A.SeasonYear = '{seasonYear}'
    AND A.SeasonType = '{seasonType}'
    AND (TE.Eventtype = 'throws' OR TE.Eventtype = 'jumps' OR TE.Eventtype = 'combined')
GROUP BY P.EventID, TE.EventName, TE.Eventtype, Ath.gender, P.AthleteSeasonID, A.schoolid,
Ath.athletefirstname, Ath.athletelastname, Ath.athleteid
```

QUERY 4: Season Best Predictions - Relays
PURPOSE: Gets season best relay performances
SQL:

```
SELECT
    P.EventID,
    TE.EventName,
    TE.Eventtype,
    '{gender}' AS gender,
```

```
    AthS.schoolid,
    AthS.schoolid AS athletefirstname,
    ' ' AS athletelastname,
    'Unavailable' AS athleteid,
    MIN(P.Resultvalue) AS predictedresult
FROM Performance AS P
JOIN CentennialConferenceEvents AS CCE ON P.EventID = CCE.EventID
JOIN RelayTeamMembers AS RTM ON P.RelayTeamID = RTM.RelayTeamID
JOIN AthleteSeason AthS ON RTM.AthleteSeasonID = AthS.AthleteSeasonID
JOIN Athlete AS Ath ON AthS.AthleteID = Ath.AthleteID
JOIN TrackEvent AS TE ON P.EventID = TE.EventID
WHERE Ath.Gender = '{gender}'
    AND AthS.SeasonYear = '{seasonYear}'
    AND AthS.SeasonType = '{seasonType}'
GROUP BY P.EventID, TE.EventName, TE.Eventtype, AthS.schoolid
```

QUERY 5: Average Season Performance - Individual Events
PURPOSE: Gets average performance per athlete per event for the season
SQL:

```
SELECT
    P.EventID,
    TE.EventName,
    TE.Eventtype,
    Ath.gender,
    P.AthleteSeasonID,
    A.schoolid,
    Ath.athletefirstname,
    Ath.athletelastname,
    Ath.athleteid,
    AVG(P.Resultvalue) AS predictedresult
FROM Performance AS P
JOIN CentennialConferenceEvents AS CCE ON P.EventID = CCE.EventID
JOIN AthleteSeason AS A ON P.AthleteSeasonID = A.AthleteSeasonID
JOIN Athlete AS Ath ON A.AthleteID = Ath.AthleteID
JOIN TrackEvent AS TE ON P.EventID = TE.EventID
WHERE Ath.Gender = '{gender}'
    AND A.SeasonYear = '{seasonYear}'
    AND A.SeasonType = '{seasonType}'
GROUP BY P.EventID, TE.EventName, TE.Eventtype, Ath.gender, P.AthleteSeasonID, A.schoolid,
Ath.athletefirstname, Ath.athletelastname, Ath.athleteid
```

QUERY 6: Average Season Performance - Relays
PURPOSE: Gets average relay performance per school per event
SQL:

```
SELECT
    P.EventID,
    TE.EventName,
    TE.Eventtype,
    '{gender}' AS gender,
```

```
    AthS.schoolid,
    AthS.schoolid AS athletefirstname,
    ' ' AS athletelastname,
    'Unavailable' AS athleteid,
    AVG(P.Resultvalue) AS predictedresult
FROM Performance AS P
JOIN CentennialConferenceEvents AS CCE ON P.EventID = CCE.EventID
JOIN RelayTeamMembers AS RTM ON P.RelayTeamID = RTM.RelayTeamID
JOIN AthleteSeason AthS ON RTM.AthleteSeasonID = AthS.AthleteSeasonID
JOIN Athlete AS Ath ON AthS.AthleteID = Ath.AthleteID
JOIN TrackEvent AS TE ON P.EventID = TE.EventID
WHERE Ath.Gender = '{gender}'
    AND AthS.SeasonYear = '{seasonYear}'
    AND AthS.SeasonType = '{seasonType}'
GROUP BY P.EventID, TE.EventName, TE.Eventtype, AthS.schoolid

QUERY 7: Linear Regression Predictions
PURPOSE: Retrieves pre-calculated linear regression predictions from analysis table
SQL:
SELECT
    EventID,
    EventName,
    EventType,
    Gender,
    SchoolID,
    AthleteID,
    AthleteFirstName,
    AthleteLastName,
    predictedresult
FROM LinearRegressionPredictions
WHERE Gender = '{gender}'
    AND SeasonType = '{seasonType}'
    AND SeasonYear = '{seasonYear}'
```

7. REFERENCE QUERIES (db_generating/queries.sql)

This file contains 20 example queries demonstrating various database operations.
These are reference queries for Phase 1 requirements, not used in the application.

```
QUERY 1: Personal Best (General)
PURPOSE: Find personal-best time for a specific athlete in a specific event
SQL:
SELECT
    a.AthleteFirstName,
    a.AthleteLastName,
    e.EventName,
    MIN(p.ResultValue) AS PersonalBest
FROM Performance p
```

```
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE a.AthleteLastName = 'Ye'
   AND a.AthleteFirstName = 'Spencer'
   AND e.EventName LIKE '%400%'
   AND e.IsRelay = FALSE
GROUP BY a.AthleteFirstName, a.AthleteLastName, e.EventName
```

QUERY 2: Athletes from a School in a Season
PURPOSE: List athletes from a specific school in a specific season
SQL:

```
SELECT DISTINCT
   a.AthleteFirstName,
   a.AthleteLastName,
   ats.ClassYear
FROM Athlete a
JOIN AthleteSeason ats ON a.AthleteID = ats.AthleteID
WHERE ats.SchoolID = 'Johns_Hopkins'
   AND ats.SeasonYear = 2025
   AND ats.SeasonType = 'Indoor'
ORDER BY a.AthleteLastName, a.AthleteFirstName
```

QUERY 3: Season Bests for an Athlete
PURPOSE: Get season bests for a specific athlete in each season
SQL:

```
SELECT
   ats.SeasonYear,
   ats.SeasonType,
   e.EventName,
   MIN(p.ResultValue) AS SeasonBest
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE a.AthleteLastName = 'Colletti'
   AND a.AthleteFirstName = 'Alex'
   AND e.EventName LIKE '%200%'
   AND e.IsRelay = FALSE
GROUP BY ats.SeasonYear, ats.SeasonType, e.EventName
ORDER BY ats.SeasonYear DESC, ats.SeasonType
```

QUERY 4: All Results for an Athlete in a Year
PURPOSE: Get all results for a specific athlete in a specific year
SQL:

```
SELECT
   tm.MeetName,
   tm.StartDate,
   e.EventName,
```

```
    p.ResultValue,
    p.WindGauge
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE a.AthleteLastName = 'Ye'
    AND a.AthleteFirstName = 'Spencer'
    AND ats.SeasonYear = 2025
ORDER BY tm.StartDate, e.EventName

QUERY 5: Conference Championship Results
PURPOSE: Get top performers in a specific event at Conference Championships
SQL:
SELECT
    a.AthleteFirstName,
    a.AthleteLastName,
    s.SchoolName,
    p.ResultValue
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE e.EventName LIKE '%800%'
    AND e.IsRelay = FALSE
    AND tm.MeetName ILIKE '%centennial%conference%'
    AND EXTRACT(YEAR FROM tm.StartDate) = 2024
ORDER BY p.ResultValue ASC
LIMIT 10

QUERY 6: Team Scoring at Conference Championships
PURPOSE: Calculate team points using 10-8-6-5-4-3-2-1 scoring
SQL:
WITH RankedPerformances AS (
    SELECT
        p.PerformanceID,
        e.EventID,
        e.EventName,
        COALESCE(ats.SchoolID, rt.SchoolID) AS SchoolID,
        p.ResultValue,
        CASE
            WHEN e.MeasureUnit = 'seconds' THEN
                ROW_NUMBER() OVER (PARTITION BY e.EventID ORDER BY p.ResultValue ASC)
            ELSE
                ROW_NUMBER() OVER (PARTITION BY e.EventID ORDER BY p.ResultValue DESC)
        END AS Place
```

```
    FROM Performance p
    JOIN TrackEvent e ON p.EventID = e.EventID
    JOIN TrackMeet tm ON p.MeetID = tm.MeetID
    LEFT JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
    LEFT JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
    WHERE tm.MeetName ILIKE '%centennial%conference%outdoor%'
        AND EXTRACT(YEAR FROM tm.StartDate) = 2024
)
SELECT
    s.SchoolName,
    SUM(
        CASE Place
            WHEN 1 THEN 10
            WHEN 2 THEN 8
            WHEN 3 THEN 6
            WHEN 4 THEN 5
            WHEN 5 THEN 4
            WHEN 6 THEN 3
            WHEN 7 THEN 2
            WHEN 8 THEN 1
            ELSE 0
        END
    ) AS TotalPoints
FROM RankedPerformances rp
JOIN School s ON rp.SchoolID = s.SchoolID
WHERE rp.Place <= 8
GROUP BY s.SchoolName
ORDER BY TotalPoints DESC

QUERY 7: Top Performers by Event Across Conference
PURPOSE: Get top 5 performers in each event for a season
SQL:
WITH RankedAthletes AS (
    SELECT
        a.AthleteFirstName,
        a.AthleteLastName,
        s.SchoolName,
        e.EventName,
        MIN(p.ResultValue) AS BestMark,
        ROW_NUMBER() OVER (
            PARTITION BY e.EventID
            ORDER BY MIN(p.ResultValue) ASC
        ) AS ConferenceRank
    FROM Performance p
    JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
    JOIN Athlete a ON ats.AthleteID = a.AthleteID
    JOIN School s ON ats.SchoolID = s.SchoolID
    JOIN TrackEvent e ON p.EventID = e.EventID
    WHERE ats.SeasonYear = 2025
```

```
        AND ats.SeasonType = 'Indoor'
        AND e.IsRelay = FALSE
        AND e.MeasureUnit = 'seconds'
    GROUP BY a.AthleteID, a.AthleteFirstName, a.AthleteLastName, s.SchoolName, e.EventID,
e.EventName
)
SELECT EventName, AthleteFirstName, AthleteLastName, SchoolName, BestMark, ConferenceRank
FROM RankedAthletes
WHERE ConferenceRank <= 5
ORDER BY EventName, ConferenceRank
```

QUERY 8: Athletes Who Competed in Multiple Events at Same Meet
PURPOSE: Find athletes who ran both individual and relay at Penn Relays
SQL:

```
SELECT DISTINCT
    a.AthleteFirstName,
    a.AthleteLastName,
    s.SchoolName
FROM Athlete a
JOIN AthleteSeason ats ON a.AthleteID = ats.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
WHERE
    EXISTS (
        SELECT 1 FROM Performance p
        JOIN TrackMeet tm ON p.MeetID = tm.MeetID
        WHERE p.AthleteSeasonID = ats.AthleteSeasonID
            AND tm.MeetName ILIKE '%penn relays%'
    )
    AND
    EXISTS (
        SELECT 1 FROM RelayTeamMembers rtm
        JOIN RelayTeam rt ON rtm.RelayTeamID = rt.RelayTeamID
        JOIN TrackMeet tm ON rt.MeetID = tm.MeetID
        WHERE rtm.AthleteSeasonID = ats.AthleteSeasonID
            AND tm.MeetName ILIKE '%penn relays%'
    )
```

QUERY 9: Athlete with Most Relay Appearances
PURPOSE: Find athletes who have competed in the most relays
SQL:

```
SELECT
    a.AthleteFirstName,
    a.AthleteLastName,
    s.SchoolName,
    COUNT(DISTINCT rtm.RelayTeamID) AS RelayCount
FROM RelayTeamMembers rtm
JOIN AthleteSeason ats ON rtm.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
```

```
GROUP BY a.AthleteID, a.AthleteFirstName, a.AthleteLastName, s.SchoolName
ORDER BY RelayCount DESC
LIMIT 10
```

QUERY 10: Seniors by School
PURPOSE: Count seniors from each school in a given year
SQL:

```
SELECT
    s.SchoolName,
    COUNT(DISTINCT a.AthleteID) AS SeniorCount
FROM AthleteSeason ats
JOIN School s ON ats.SchoolID = s.SchoolID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
WHERE ats.ClassYear = 'SR'
    AND ats.SeasonYear = 2025
GROUP BY s.SchoolID, s.SchoolName
ORDER BY SeniorCount DESC
```

QUERY 11: School Roster
PURPOSE: Get full roster for a school in a season
SQL:

```
SELECT
    a.AthleteFirstName,
    a.AthleteLastName,
    ats.ClassYear,
    a.Gender
FROM Athlete a
JOIN AthleteSeason ats ON a.AthleteID = ats.AthleteID
WHERE ats.SchoolID = 'Johns_Hopkins'
    AND ats.SeasonYear = 2025
    AND ats.SeasonType = 'Indoor'
ORDER BY a.AthleteLastName, a.AthleteFirstName
```

QUERY 12: Meet Results by Event
PURPOSE: Get all results for a specific event at a specific meet
SQL:

```
SELECT
    a.AthleteFirstName,
    a.AthleteLastName,
    s.SchoolName,
    p.ResultValue,
    p.WindGauge
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE e.EventName LIKE '%60 Meters%'
```

```
    AND tm.MeetName ILIKE '%black%blue%'
ORDER BY p.ResultValue ASC
```

QUERY 13: Personal Best Progression
PURPOSE: Track how an athlete's PB has improved over seasons
SQL:

```
SELECT
    ats.SeasonYear,
    ats.SeasonType,
    ats.ClassYear,
    MIN(p.ResultValue) AS SeasonBest
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE a.AthleteLastName = 'Colletti'
    AND a.AthleteFirstName = 'Alex'
    AND e.EventName LIKE '%60 Meters%'
    AND e.IsRelay = FALSE
GROUP BY ats.SeasonYear, ats.SeasonType, ats.ClassYear
ORDER BY ats.SeasonYear, ats.SeasonType
```

QUERY 14: Event Records by School
PURPOSE: Get best performance in each event for a school
SQL:

```
SELECT
    e.EventName,
    a.AthleteFirstName,
    a.AthleteLastName,
    MIN(p.ResultValue) AS SchoolRecord
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE ats.SchoolID = 'Johns_Hopkins'
    AND e.IsRelay = FALSE
    AND e.MeasureUnit = 'seconds'
GROUP BY e.EventID, e.EventName, a.AthleteID, a.AthleteFirstName, a.AthleteLastName
ORDER BY e.EventName
```

QUERY 15: Meets Attended by School
PURPOSE: List meets a school attended in a season
SQL:

```
SELECT DISTINCT
    tm.MeetName,
    tm.StartDate,
    tm.EndDate,
    COUNT(DISTINCT p.PerformanceID) AS PerformanceCount
FROM TrackMeet tm
```

```
JOIN Performance p ON tm.MeetID = p.MeetID
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
WHERE ats.SchoolID = 'Johns_Hopkins'
   AND ats.SeasonYear = 2025
GROUP BY tm.MeetID, tm.MeetName, tm.StartDate, tm.EndDate
ORDER BY tm.StartDate
```

QUERY 16: Head-to-Head Comparison
PURPOSE: Compare two athletes in the same event
SQL:

```
SELECT
   a.AthleteFirstName || ' ' || a.AthleteLastName AS Athlete,
   tm.MeetName,
   tm.StartDate,
   p.ResultValue
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE (a.AthleteLastName = 'Colletti' OR a.AthleteLastName = 'Ye')
   AND e.EventName LIKE '%60 Meters%'
   AND e.IsRelay = FALSE
ORDER BY tm.StartDate, p.ResultValue
```

QUERY 17: Best Relay Teams
PURPOSE: Rank relay teams by performance
SQL:

```
SELECT
   s.SchoolName,
   e.EventName,
   tm.MeetName,
   p.ResultValue
FROM Performance p
JOIN RelayTeam rt ON p.RelayTeamID = rt.RelayTeamID
JOIN School s ON rt.SchoolID = s.SchoolID
JOIN TrackEvent e ON p.EventID = e.EventID
JOIN TrackMeet tm ON p.MeetID = tm.MeetID
WHERE e.EventName LIKE '%4 x 400%'
ORDER BY p.ResultValue ASC
LIMIT 10
```

QUERY 18: Athletes with Most Performances
PURPOSE: Find athletes who competed the most in a season
SQL:

```
SELECT
   a.AthleteFirstName,
   a.AthleteLastName,
   s.SchoolName,
```

```
    COUNT(*) AS PerformanceCount
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
WHERE ats.SeasonYear = 2025
    AND ats.SeasonType = 'Indoor'
GROUP BY a.AthleteID, a.AthleteFirstName, a.AthleteLastName, s.SchoolName
ORDER BY PerformanceCount DESC
LIMIT 20
```

QUERY 19: Multi-Event Athletes
PURPOSE: Find athletes who compete in 3+ different events
SQL:

```
SELECT
    a.AthleteFirstName,
    a.AthleteLastName,
    s.SchoolName,
    COUNT(DISTINCT e.EventID) AS EventCount,
    STRING_AGG(DISTINCT e.EventName, ', ') AS Events
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN Athlete a ON ats.AthleteID = a.AthleteID
JOIN School s ON ats.SchoolID = s.SchoolID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE ats.SeasonYear = 2025
    AND e.IsRelay = FALSE
GROUP BY a.AthleteID, a.AthleteFirstName, a.AthleteLastName, s.SchoolName
HAVING COUNT(DISTINCT e.EventID) >= 3
ORDER BY EventCount DESC
```

QUERY 20: Average Performance by Event Type
PURPOSE: Calculate average time/distance by event type for each school
SQL:

```
SELECT
    s.SchoolName,
    e.EventType,
    COUNT(*) AS PerformanceCount,
    ROUND(AVG(p.ResultValue), 2) AS AvgResult
FROM Performance p
JOIN AthleteSeason ats ON p.AthleteSeasonID = ats.AthleteSeasonID
JOIN School s ON ats.SchoolID = s.SchoolID
JOIN TrackEvent e ON p.EventID = e.EventID
WHERE ats.SeasonYear = 2025
GROUP BY s.SchoolID, s.SchoolName, e.EventType
ORDER BY s.SchoolName, e.EventType
```