



## Data Dictionary

### Competition Table

| Field Name            | Constraints                          | Possible Values / Description              | Data Type    |
|-----------------------|--------------------------------------|--|--------------|
| competition ID        | <b>PK</b> , NOT NULL, AUTO_INCREMENT | Unique ID for each championship            | INT          |
| competition Title     | NOT NULL, UNIQUE                     | e.g., "National Archery Championship 2025" | VARCHAR(100) |
| competition StartDate | NOT NULL                             | Date championship begins                   | DATE         |
| competition EndDate   | NOT NULL                             | Date championship ends                     | DATE         |
| competition Location  | NOT NULL                             | Venue or city name                         | VARCHAR(100) |
| championships pID     | NULLABLE                             |  | INT          |

### Championship Table

| Field Name        | Constraints                          | Possible Values / Description   | Data Type |
|-------------------|--------------------------------------|---------------------------------|-----------|
| championships pID | <b>PK</b> , NOT NULL, AUTO_INCREMENT | Unique ID for each championship | INT       |

|                              |                     |  |              |
|------------------------------|---------------------|--|--------------|
| <b>championshiptitle</b>     | NOT NULL,<br>UNIQUE | e.g., “National Archery Championship 2025” | VARCHAR(100) |
| <b>championshiptitleYear</b> | NOT NULL            | Year of Championship                       | YEAR         |

## Round Table

| Field Name           | Constraints  | Possible Values / Description         | Data Type   |
|----------------------|--|---------------------------------------|-------------|
| <b>roundID</b>       | <b>PK</b> , NOT NULL, AUTO_INCREMENT                     | Unique ID                             | INT         |
| <b>competitionID</b> | <b>FK</b> → <b>competition(competitionID)</b> , NOT NULL | The competition this round belongs to | INT         |
| <b>roundType</b>     | <b>FK</b> → <b>roundType(roundType)</b> , NOT NULL       | Round format used                     | VARCHAR(50) |
| <b>roundDate</b>     | NOT NULL   | Date of round                         | DATE        |

## RoundType Table

| Field Name       | Constraints          | Possible Values / Description      | Data Type   |
|------------------|----------------------|------------------------------------|-------------|
| <b>roundType</b> | <b>PK</b> , NOT NULL | e.g., “WA70”, “WA1440”, “Indoor18” | VARCHAR(50) |

|                         |   |                                  |             |
|-------------------------|---|----------------------------------|-------------|
| <b>baseRoundID</b><br>D | <b>FK → roundType(roundType),</b><br>NULLABLE | Used for grouping related rounds | VARCHAR(50) |
|-------------------------|---|----------------------------------|-------------|

## Range Table

| Field Name                    | Constraints  | Possible Values / Description | Data Type                                     |
|-------------------------------|--|-------------------------------|---|
| <b>rangeID</b>                | <b>PK, NOT NULL,</b><br><b>AUTO_INCREMENT</b>        | Unique ID                     | INT   |
| <b>roundType</b>              | <b>FK → roundType(roundType),</b><br><b>NOT NULL</b> | Round type associated         | VARCHAR(50)                                   |
| <b>rangeDistance</b>          | NOT NULL   | e.g., 10, 20, 30, 50, 70, 90  | ENUM('10','20','30','40','50','60','70','90') |
| <b>rangeTargetSize</b>        | NOT NULL   | Target face size (cm)         | ENUM('80','122')                              |
| <b>rangeTotalEnds</b>         | NOT NULL   | Number of ends in this range  | INT   |
| <b>rangeTotalArrowsPerEnd</b> | NOT NULL   | e.g., 6 arrows per end        | INT   |

## Archer Table

| Field Name | Constraints | Possible Values / Description | Data Type |
|------------|-------------|-------------------------------|-----------|
|------------|-------------|-------------------------------|-----------|

|                               |   |                                   |                               |
|-------------------------------|---|-----------------------------------|-------------------------------|
| <b>archerID</b>               | <b>PK</b> , NOT NULL,<br>AUTO_INCREMENT | Unique ID                         | INT                           |
| <b>archerFirst<br/>Name</b>   | NOT NULL                                | Archer's first<br>name            | VARCHAR(50)                   |
| <b>archerLast<br/>Name</b>    | NOT NULL                                | Archer's last<br>name             | VARCHAR(50)                   |
| <b>archerGender</b>           | NOT NULL                                | 'Male' ,<br>'Female' ,<br>'Mixed' | ENUM('Male','Female','Mixed') |
| <b>archerDate<br/>OfBirth</b> | NOT NULL                                | Date of birth                     | DATE                          |
| <b>archerEmail</b>            | NOT NULL                                | Email for<br>registration         | VARCHAR(100)                  |
| <b>archerPhone</b>            | NOT NULL                                | Phone no                          | VARCHAR(50)                   |
| <b>archerNationality</b>      | NOT NULL                                | Nationality                       | VARCHAR(50)                   |

### Category Table

| Field Name        | Constraints                             | Possible Values /<br>Description | Data Type                           |
|-------------------|---|----------------------------------|-------------------------------------|
| <b>categoryID</b> | <b>PK</b> , NOT NULL,<br>AUTO_INCREMENT | Unique ID                        | INT                                 |
| <b>ageGroup</b>   | NOT NULL                                | 'Under 18' ,<br>'Under 21'       | ENUM('U18','U21','Senior','Master') |

|           |          |   |  |
|-----------|----------|---|--|
|           |          | '21' ,<br>'Senior' ,<br>'Master'            |  |
| gender    | NOT NULL | 'Male' ,<br>'Female' ,<br>'Mixed'           | ENUM('Male','Female')                    |
| equipment | NOT NULL | 'Recurve' ,<br>'Compound'<br>'<br>'Barebow' | ENUM('Recurve','Compound','Bar<br>ebow') |

RoundCategory Table

| Field Name | Constraints                                       | Possible Values / Description | Data Type |
|------------|---|-------------------------------|-----------|
| roundID    | <b>PK, FK</b> → round(roundID),<br>NOT NULL       | Round                         | INT       |
| categoryID | <b>PK, FK</b> → category(categoryID), NOT<br>NULL | Category                      | INT       |

ParticipationCategory Table

| Field Name      | Constraints   | Possible Values / Description | Data Type |
|-----------------|---|-------------------------------|-----------|
| participationID | <b>PK, FK</b> → participation(participationID),<br>NOT NULL | participationID               | INT       |

|                   |  |          |     |
|-------------------|--|----------|-----|
| <b>categoryID</b> | <b>PK, FK → category(categoryID), NOT NULL</b> | Category | INT |
|-------------------|--|----------|-----|

### Participation Table

| Field Name             | Constraints                                      | Possible Values / Description | Data Type |
|------------------------|--|-------------------------------|-----------|
| <b>participationID</b> | <b>PK, NOT NULL, AUTO_INCREMENT</b>              | Unique participation record   | INT       |
| <b>archerID</b>        | <b>FK → archer(archerID), NOT NULL</b>           | Archer participating          | INT       |
| <b>competitionID</b>   | <b>FK → competition(competitionID), NOT NULL</b> | Competition joined            | INT       |

### ArrowStaging Table

| Field Name             | Constraints  | Possible Values / Description | Data Type |
|------------------------|--|-------------------------------|-----------|
| <b>arrowStagingID</b>  | <b>PK, AUTO_INCREMENT</b>                            | Unique ID                     | INT       |
| <b>roundID</b>         | <b>FK → round(roundID), NOT NULL</b>                 | Round                         | INT       |
| <b>participationID</b> | <b>FK → participation(participationID), NOT NULL</b> | Archer in competition         | INT       |

|                         |                                      |  |                                      |
|-------------------------|--------------------------------------|--|--------------------------------------|
| <b>distance</b>         | <b>FK → range(rangeID), NOT NULL</b> | Which range of the round                 | INT                                  |
| <b>endOrder</b>         | NOT NULL                             | 1...N (for each end)                     | INT                                  |
| <b>arrowScore</b>       | NOT NULL                             | arrow score (0 - 10)                     | INT                                  |
| <b>isX</b>              | NOT NULL                             | is inner 10 mark                         | BOOL                                 |
| <b>arrowStageStatus</b> | NOT NULL                             | whether a score is confirmed by recorder | ENUM('confirm', 'pending', 'reject') |

### RoundScore Table

| Field Name             | Constraints  | Possible Values / Description      | Data Type |
|------------------------|--|------------------------------------|-----------|
| <b>roundScore ID</b>   | <b>PK, AUTO_INCREMENT</b>                            | Unique ID                          | INT       |
| <b>participationID</b> | <b>FK → participation(participationID), NOT NULL</b> | Link to archer in that competition | INT       |
| <b>roundID</b>         | <b>FK → round(roundID), NOT NULL</b>                 | The round being scored             | INT       |
| <b>totalScore</b>      | NOT NULL   | e.g., 654                          | INT       |
| <b>totalX</b>          | NOT NULL   | Number of “X” hits (bullseye)      | INT       |
| <b>totalTen</b>        | NOT NULL   | Number of 10s                      | INT       |

|                   |          |                          |      |
|-------------------|----------|--------------------------|------|
| <b>dateRecord</b> | NOT NULL | Date of the shooting day | DATE |
| <b>ed</b>         |          |                          |      |

### ChampionshipResult Table

| Field Name            | Constraints                                  | Possible Values / Description | Data Type                        |
|-----------------------|--|-------------------------------|----------------------------------|
| <b>archerID</b>       | <b>PK, FK → archer(archerID)</b>             | Archer                        | INT                              |
| <b>championshipID</b> | <b>PK, FK → championship(championshipID)</b> | Championship                  | INT                              |
| <b>totalScore</b>     | NOT NULL                                     | Sum of all rounds             | INT                              |
| <b>ageGroup</b>       | NOT NULL                                     | age group                     | ENUM('Open', 'Under 21', etc)    |
| <b>equipment</b>      | NOT NULL                                     | equipment                     | ENUM('Recurve', 'compound', etc) |
| <b>gender</b>         | NOT NULL                                     | Male, Female                  | ENUM('Male', 'Female')           |



## Development Convention

### Database Design Project: Conventions and Workflow

This document outlines the standard conventions and phased workflow to be followed during the database design project, ensuring consistency, clarity, and effective data modeling.

#### 1. Naming Convention

To maintain a clean and standardized structure across the entire database schema, we will strictly adhere to the **Camel Case** naming convention for all database objects.

Convention Details:

| Database Object                    | Naming Rule  | Example   |
|------------------------------------|--|---|
| <b>Tables / Collections</b>        | Start with a lowercase letter. The first letter of each subsequent concatenated word is capitalized. Use singular nouns. | memberProfile , archerySession                      |
| <b>Attributes / Columns</b>        | Start with a lowercase letter. The first letter of each subsequent concatenated word is capitalized.                     | firstName , sessionDate , totalScore                |
| <b>Foreign Keys<br/>(Optional)</b> | Use a descriptive name derived from the parent table and attribute, following camel case.                                | memberProfileId (if linking to memberProfile table) |

**Key Principle:** Use descriptive, meaningful names that clearly indicate the object's purpose and content. **Avoid underscores, hyphens, and spaces.**

#### 2. Project Workflow: Iterative Design and Integration

The project will follow a structured, three-phase workflow. The core principle is to solve each requirement in isolation before combining them into a final, unified schema.

#### Phase 1: Decomposition and Modeling

We will address each user story independently to ensure all specific requirements are met without initial interference from other modules.

| Step                     | Action  | Deliverable   |
|--------------------------|---|---|
| <b>1.1 Analysis</b>      | Focus exclusively on a single user story:<br><b>Archery, Club Recorder, or Sys Admin.</b>   | Requirements List   |
| <b>1.2 Modeling</b>      | Develop the initial, unnormalized database structure required to fulfill the user story's requirements. This may involve redundancy or data anomalies.              | Preliminary <b>Entity-Relationship Diagram (ERD)</b> or equivalent logical model. |
| <b>1.3 Query Testing</b> | Draft necessary <b>SQL Queries</b> (or pseudo-queries) to ensure the model can support the core functionality (e.g., adding a score, updating a membership record). | Sample Queries  |

#### Phase 2: Normalization

For each individual user story model developed in Phase 1, we will systematically apply normalization principles to eliminate data redundancy and insertion/update/deletion anomalies.

| Step | Goal | Technique |
|------|------|-----------|
|------|------|-----------|

|                                     |  |  |
|-------------------------------------|--|--|
| <b>2.1 First Normal Form (1NF)</b>  | Ensure all column values are atomic (indivisible) and eliminate repeating groups.  | Decompose repeating groups into separate tables. |
| <b>2.2 Second Normal Form (2NF)</b> | Ensure all non-key attributes are fully functionally dependent on the primary key.<br>(Applicable only to composite keys). | Isolate partial dependencies into new tables.    |
| <b>2.3 Third Normal Form (3NF)</b>  | Eliminate transitive dependencies, ensuring non-key attributes are only dependent on the primary key and nothing else.     | Isolate transitive dependencies into new tables. |

The output of this phase will be three separate, fully normalized schemas (one for Archery, one for Club Recorder, and one for Sys Admin).

#### Phase 3: Integration and Final Schema

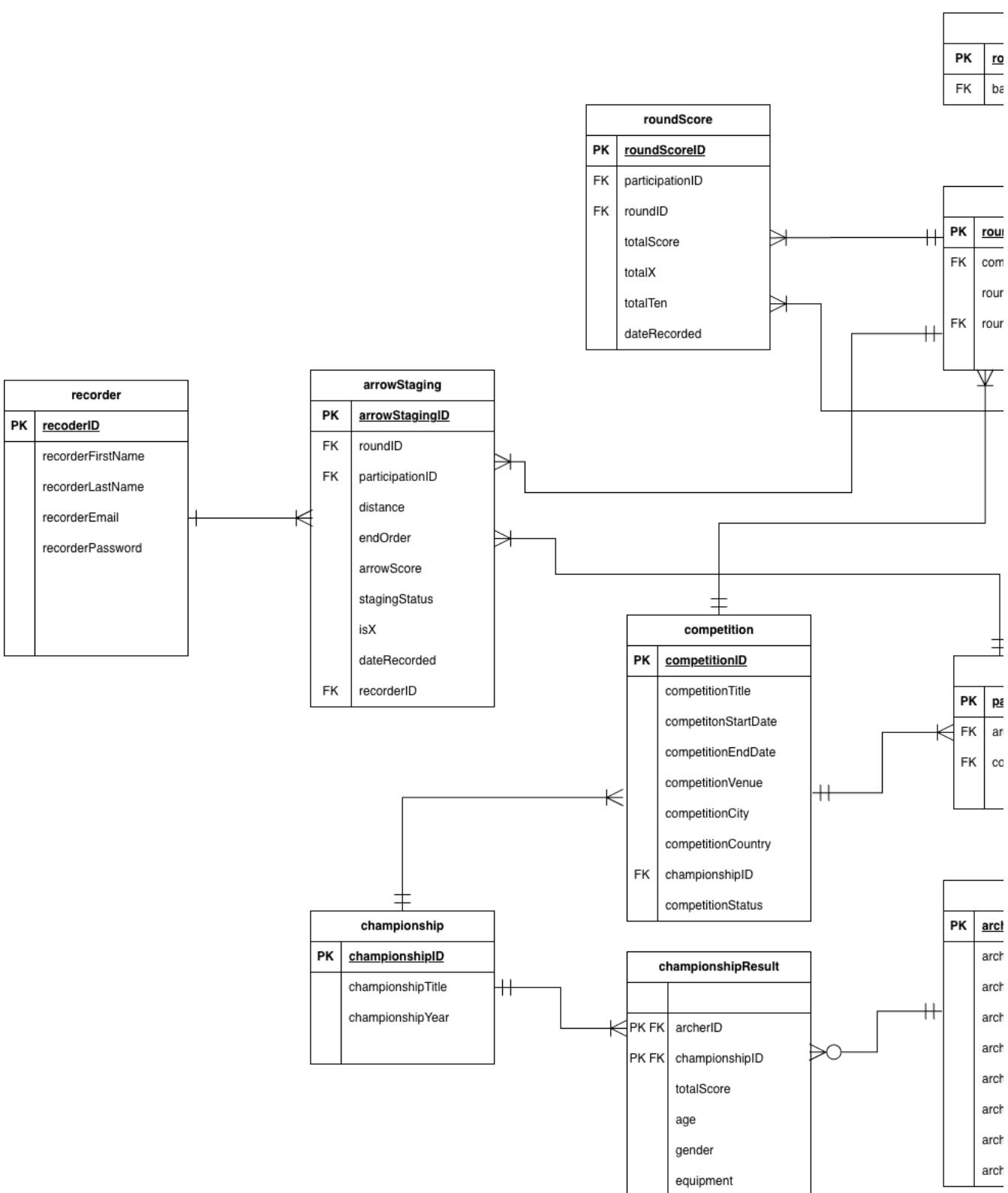
The final step is to merge the three normalized schemas into a single, cohesive database design.

| Step                          | Action  | Outcome   |
|-------------------------------|---|---|
| <b>3.1 Key Identification</b> | Identify common entities and potential primary/foreign key relationships between the three normalized models. | Defined relationships (e.g., how the <b>Member</b> entity in Archery relates to the <b>User</b> entity in Sys Admin). |
| <b>3.2 Schema Merge</b>       | Combine the tables from the three models,   | A single, unified, post-normalization ERD.  |

|                                  |   |                               |
|----------------------------------|---|-------------------------------|
|                                  | <p>resolving any naming conflicts (using the agreed-upon camel case standard) and eliminating redundant entities.</p>                                 |                               |
| <b>3.3 Review and Validation</b> | <p>Validate the final schema against all original user story requirements to ensure functionality is maintained and the structure adheres to 3NF.</p> | Final Database Schema Design. |



ERD Diagram





## Database Indexing Recommendations

archer

**Primary Key (Already indexed):** archerID

**Additional Indexes Needed:**

- Email column for authentication lookups
- Gender and dateOfBirth combination for category filtering

**Reasoning:** Authentication queries will frequently search by email. Filtering archers by gender and age categories (derived from dateOfBirth) is common for competition management and reporting.

```
1 -- Email index for authentication
2 CREATE INDEX idx_archer_email ON archer(archerEmail);
3
4 -- Composite index for category filtering (gender + date of birth)
5 CREATE INDEX idx_archer_gender_dob ON archer(archerGender,
6 archerDateOfBirth);
```

---

arrowStaging

**Primary Key (Already indexed):** arrowStagingID

**Foreign Keys (Already indexed):** roundID, participationID, recorderID

category

**Primary Key (Already indexed):** categoryID

**Additional Indexes Needed:**

- Composite index on equipment + age + gender for category lookups

**Reasoning:** Categories are always searched by their combination of equipment, age, and gender. This is the natural key for category lookups.

```
1 -- Composite index for category lookup
2 CREATE INDEX idx_category_equipment_age_gender ON category(equipment,
3 ageGroup, gender);
```

---

championship

**Primary Key (Already indexed):** championshipID

championshipResult

**Primary Key (Already indexed):** archerID, championshipID (composite)

**Foreign Key (Already indexed):** championshipID

competition

**Primary Key (Already indexed):** competitionID

**Foreign Key (Already indexed):** championshipID

participation

**Primary Key (Already indexed):** participationID

**Foreign Keys (Already indexed):** archerID , competitionID

**Additional Indexes Needed:**

- Composite index on competitionID + archerID for competition participant lookups

**Reasoning:** The most common query pattern is finding all participants in a competition or checking if an archer is registered for a competition. The existing FK indexes cover basic joins, but a composite in this order optimizes the primary query pattern.

```
1 | -- Composite index for competition participant queries
2 | CREATE INDEX idx_participation_comp_archer ON
3 | participation(competitionID, archerID);
```

---

participationCategory

**Primary Key (Already indexed):** participationID , categoryID (composite)

**Foreign Key (Already indexed):** categoryID

**Additional Indexes Needed:** None

**Reasoning:** The existing composite primary key and FK index are sufficient. This is a many-to-many junction table where both FK indexes already exist, covering all typical query patterns.

```
1 | -- No additional indexes needed
2 |
```

---

range

**Primary Key (Already indexed):** rangeID

**Foreign Key (Already indexed):** roundType

recorder

**Primary Key (Already indexed):** recorderID

**Unique Key (Already indexed):** recordEmail

**Additional Indexes Needed:** None

**Reasoning:** The unique email index already exists for authentication. This small lookup table doesn't require additional indexes.

```
1 | -- No additional indexes needed
2 |
```

---

## round

**Primary Key (Already indexed):** roundID

**Foreign Keys (Already indexed):** competitionID, roundType

## roundCategory

**Primary Key (Already indexed):** roundID, categoryID (composite)

**Foreign Key (Already indexed):** categoryID

**Additional Indexes Needed:** None

**Reasoning:** The existing composite PK and FK index cover all typical query patterns for this junction table.

```
1 | -- No additional indexes needed
2 |
```

---

## roundScore

**Primary Key (Already indexed):** roundScoreID

**Foreign Keys (Already indexed):** participationID, roundID

**Additional Indexes Needed:**

- Composite index on roundID + totalScore for round leaderboards
- Composite index on participationID + dateRecorded for archer score history

**Reasoning:** Round leaderboards require efficient ordering by score within a round. Archer score history queries need chronological ordering of scores for a participation.

```
1 | -- Composite index for round leaderboard queries
2 | CREATE INDEX idx_roundscore_round_score ON roundScore(roundID,
totalScore DESC);
3 |
4 | -- Composite index for archer score history
5 | CREATE INDEX idx_roundscore_participation_date ON
roundScore(participationID, dateRecorded);
6 |
```

---

## roundType

**Primary Key (Already indexed):** roundType

**Foreign Key (Already indexed):** baseRoundID

**Additional Indexes Needed:** None

**Reasoning:** This is a small reference table. The existing PK and FK index are sufficient for all lookups and hierarchical queries.

```
1 | -- No additional indexes needed
2 |
```

---

## Summary of Indexing Strategy

**Total New Indexes to Create: 14**

**Tables Requiring New Indexes:** 10 out of 14 tables

**Key Principles Applied:**

1. **Authentication columns** (email) are always indexed
2. **Common filter combinations** get composite indexes
3. **Sorting columns** in leaderboard queries are indexed
4. **Date/temporal columns** used for filtering are indexed
5. **Junction tables** with existing composite PKs and FKs don't need additional indexes
6. **Small lookup tables** don't need additional indexes beyond PKs

**Performance Impact:**

- **High Impact:** archer email, competition date/status, roundScore leaderboard indexes
- **Medium Impact:** category composite, participation composite, arrowStaging composite
- **Low Impact:** championship year, range composite, round date composite

**Maintenance Considerations:**

- Indexes will slightly slow down INSERT/UPDATE/DELETE operations
- All recommended indexes target high-frequency read operations
- Monitor index usage and remove unused indexes after deployment

## Query (3 Queries per person)

### Archery Competition Database Documentation

#### Database Queries

##### Query 1: Get All Archers with Their Competition Participations (Show Wai Yan)

**Purpose:** Retrieve a list of all archers along with their participation details in competitions. This is needed for registration verification, attendance tracking, and competitor management.

##### Why it's needed:

- Competition organizers need to track who is participating in which events
- Helps in generating participant lists and badges
- Essential for attendance verification and competition planning

```
1   SELECT
2       a.archerID,
3       a.archerFirstName,
4       a.archerLastName,
5       a.archerGender,
6       a.archerNationality,
7       c.title AS competitionTitle,
8       c.location,
9       c.startDate,
10      c.endDate,
11      p.participationID
12  FROM archer a
13  JOIN participation p ON a.archerID = p.archerID
14  JOIN competition c ON p.competitionID = c.competitionID
15  ORDER BY a.archerLastName, c.startDate;
16
```

---

##### Query 2: Get Archer Contact Information and Participation History (Show Wai Yan)

**Purpose:** Retrieve archer contact details along with their competition participation history. Essential for communication, marketing, and membership management.

##### Why it's needed:

- Enables communication with participants for upcoming events
- Supports marketing and promotional activities
- Helps in membership renewal and engagement
- Provides data for participation statistics and retention analysis

```
1   SELECT
2       a.archerID,
3       a.archerFirstName,
4       a.archerLastName,
5       a.archerEmail,
6       a.archerPhone,
7       a.archerNationality,
8       a.archerGender,
9       YEAR(CURDATE()) - YEAR(a.archerDateOfBirth) AS currentAge,
10      COUNT(DISTINCT p.competitionID) AS totalCompetitions,
11      MAX(c.startDate) AS lastCompetitionDate,
```

```

12     MIN(c.startDate) AS firstCompetitionDate
13 FROM archer a
14 LEFT JOIN participation p ON a.archerID = p.archerID
15 LEFT JOIN competition c ON p.competitionID = c.competitionID
16 GROUP BY a.archerID
17 ORDER BY totalCompetitions DESC, a.archerLastName;
18

```

#### Query 3: Get Round Scores with Archer Details for a Specific Competition (Show Wai Yan)

**Purpose:** Retrieve detailed scoring information for all archers in a specific competition round. Critical for live scoring displays and performance analysis.

#### Why it's needed:

- Real-time score tracking during competitions
- Performance analysis for coaches and archers
- Verification of scoring accuracy
- Historical performance records

```

1  SELECT
2      a.archerFirstName,
3      a.archerLastName,
4      r.roundID,
5      r.date AS roundDate,
6      rt.roundType,
7      rs.totalScore,
8      rs.totalX,
9      rs.totalTen,
10     rs.dateRecorded
11 FROM roundScore rs
12 JOIN participation p ON rs.participationID = p.participationID
13 JOIN archer a ON p.archerID = a.archerID
14 JOIN round r ON rs.roundID = r.roundID
15 JOIN roundType rt ON r.roundType = rt.roundType
16 ORDER BY rs.totalScore DESC, rs.totalX DESC;
17

```

#### Query 4: Find Competitions by Location and Date Range (Aung Moe Thu)

**Purpose:** Search for competitions happening in specific locations within a date range. Useful for archers planning their competition schedule.

#### Why it's needed:

- Helps archers plan their competition calendar
- Assists in travel and accommodation arrangements
- Enables regional competition searches
- Supports competition coordination to avoid scheduling conflicts

```

1  SELECT
2      c.competitionID,
3      c.title,
4      c.location,
5      c.startDate,
6      c.endDate,
7      c.status,
8      ch.title AS championshipTitle,
9      COUNT(DISTINCT p.archerID) AS numberOfParticipants
10 FROM competition c
11 LEFT JOIN championship ch ON c.championshipID = ch.championshipID

```

```

12 LEFT JOIN participation p ON c.competitionID = p.competitionID
13 WHERE c.location LIKE '%Sydney%'
14     AND c.startDate BETWEEN '2024-01-01' AND '2024-12-31'
15 GROUP BY c.competitionID
16 ORDER BY c.startDate;
17

```

#### Query 5: Get Arrow-by-Arrow Scoring Details for a Specific Archer (Aung Moe Thu)

**Purpose:** Track individual arrow scores for detailed performance analysis. Essential for identifying shooting patterns and improvement areas.

#### Why it's needed:

- Detailed performance analysis for coaching
- Identifies consistency issues or patterns
- Helps in technique improvement
- Provides granular data for statistical analysis

```

1  SELECT
2      a.archerFirstName,
3      a.archerLastName,
4      r.roundID,
5      rt.roundType,
6      ars.distance,
7      ars.endOrder,
8      ars.arrowScore,
9      ars.isX,
10     ars.date AS scoredDate,
11     CONCAT(rec.recorderFirstName, ' ', rec.recorderLastName) AS
12     recorderName
13 FROM arrowStaging ars
14 JOIN participation p ON ars.participationID = p.participationID
15 JOIN archer a ON p.archerID = a.archerID
16 JOIN round r ON ars.roundID = r.roundID
17 JOIN roundType rt ON r.roundType = rt.roundType
18 LEFT JOIN recorder rec ON ars.recorderID = rec.recorderID
19 ORDER BY ars.date, ars.distance, ars.endOrder;

```

#### Query 6: List All Categories with Participant Counts for a Competition (Aung Moe Thu)

**Purpose:** Show how many archers are competing in each category for a specific competition. Important for organizing competition logistics.

#### Why it's needed:

- Determines target allocation and scheduling
- Helps organize qualification rounds
- Ensures adequate resources (judges, targets, time slots)
- Supports competition structure planning

```

1  SELECT
2      cat.categoryID,
3      cat.equipment,
4      cat.ageGroup,
5      cat.gender,
6      c.competitionTitle,
7      COUNT(DISTINCT pc.participationID) AS participantCount
8  FROM category cat
9  LEFT JOIN participationCategory pc ON cat.categoryID = pc.categoryID
10 LEFT JOIN participation p ON pc.participationID = p.participationID

```

```

11 LEFT JOIN competition c ON p.competitionID = c.competitionID
12 WHERE c.competitionID = 1
13 GROUP BY cat.categoryID, cat.equipment, cat.ageGroup, cat.gender
14 ORDER BY participantCount DESC;

```

#### Query 7: Find Top Performers by Gender and Equipment Type (Khart Thu Aung)

**Purpose:** Identify the highest-scoring archers in each gender and equipment category. Used for recognition and seeding purposes.

#### Why it's needed:

- Highlights top performers for awards and recognition
- Provides seeding information for tournaments
- Generates statistics for media and promotion
- Tracks record holders

```

1   SELECT
2       a.archerID,
3       a.archerFirstName,
4       a.archerLastName,
5       a.archerGender,
6       cat.equipment,
7       MAX(rs.totalScore) AS highestScore,
8       SUM(rs.totalXs) AS totalXs,
9       SUM(rs.totalTens) AS totalTens
10  FROM archer a
11  JOIN participation p ON a.archerID = p.archerID
12  JOIN participationCategory pc ON p.participationID =
13      pc.participationID
14  JOIN category cat ON pc.categoryID = cat.categoryID
15  JOIN roundScore rs ON p.participationID = rs.participationID
16  GROUP BY a.archerID, a.archerGender, cat.equipment
17  HAVING MAX(rs.totalScore) > 1000
18  ORDER BY cat.equipment, a.archerGender, highestScore DESC;

```

#### Query 8: Get Competition Schedule with Round Types and Dates (Khart Thu Aung)

**Purpose:** Display the complete schedule of rounds for a competition. Essential for participants and organizers to plan their activities.

#### Why it's needed:

- Participants need to know when to compete
- Helps in resource allocation (judges, equipment)
- Enables schedule conflict management
- Supports venue coordination

```

1   SELECT
2       c.competitionID,
3       c.title AS competitionTitle,
4       r.roundID,
5       r.date AS roundDate,
6       rt.roundType,
7       rt.baseRoundTypeID,
8       COUNT(DISTINCT rs.participationID) AS participantsInRound
9   FROM competition c
10  JOIN round r ON c.competitionID = r.competitionID
11  JOIN roundType rt ON r.roundType = rt.roundType
12  LEFT JOIN roundScore rs ON r.roundID = rs.roundID

```

```
13 WHERE c.competitionID = 1
14 GROUP BY r.roundID
15 ORDER BY r.date, rt.roundType;
16
```

---

#### Query 9: Analyze Archer Performance Trends Over Multiple Competitions (Khart Thu Aung)

**Purpose:** Track an archer's scoring progression across multiple competitions. Valuable for performance monitoring and training effectiveness.

##### Why it's needed:

- Monitors athlete development over time
- Identifies improvement or decline in performance
- Evaluates training program effectiveness
- Supports coaching decisions and goal setting

```
1   SELECT
2     a.archerID,
3     CONCAT(a.archerFirstName, ' ', a.archerLastName) AS archerName,
4     c.competitionTitle,
5     c.competitionStartDate,
6     AVG(rs.totalScore) AS avgScore,
7     MAX(rs.totalScore) AS maxScore,
8     MIN(rs.totalScore) AS minScore,
9     SUM(rs.totalXs) AS totalXs,
10    COUNT(rs.roundScoreID) AS roundsCompleted
11   FROM archer a
12   JOIN participation p ON a.archerID = p.archerID
13   JOIN competition c ON p.competitionID = c.competitionID
14   JOIN roundScore rs ON p.participationID = rs.participationID
15   WHERE a.archerID = 1
16   GROUP BY c.competitionID
17   ORDER BY c.competitionStartDate;
18
```

---

#### Query 10: List All Round Types with Their Distance Configurations (Kaung Htet Nyein)

**Purpose:** Display the complete range setup for each round type. Critical for competition planning and target allocation.

##### Why it's needed:

- Ensures proper range configuration before competitions
- Helps in planning equipment and space requirements
- Informs participants about competition format
- Supports compliance with archery regulations

```
1   SELECT
2     rt.roundType,
3     rt.baseRoundTypeID,
4     rg.rangeID,
5     rg.distance,
6     rg.targetSize,
7     rg.totalEnds,
8     rg.totalArrowsPerEnd,
9     (rg.totalEnds * rg.totalArrowsPerEnd) AS totalArrows
10   FROM roundType rt
11   JOIN range rg ON rt.roundType = rg.roundType
12   ORDER BY rt.roundType, rg.distance DESC;
13
```

#### Query 11: Find Archers Eligible for Multiple Categories (Kaung Htet Nyein)

**Purpose:** Identify archers who qualify for and are registered in multiple competition categories. Useful for scheduling and category management.

#### Why it's needed:

- Prevents scheduling conflicts for multi-category competitors
- Ensures adequate rest time between events
- Helps organize competition flow efficiently
- Supports fair competition management

```
1  SELECT
2      a.archerID,
3      CONCAT(a.archerFirstName, ' ', a.archerLastName) AS archerName,
4      c.competitionTitle,
5      COUNT(DISTINCT pc.categoryID) AS categoryCount,
6      GROUP_CONCAT(
7          DISTINCT CONCAT(cat.equipment, ' - ', cat.ageGroup, ' - ',
8          cat.gender)
9          SEPARATOR ';'
10     ) AS categories
11  FROM archer a
12  JOIN participation p
13    ON a.archerID = p.archerID
14  JOIN competition c
15    ON p.competitionID = c.competitionID
16  JOIN participationCategory pc
17    ON p.participationID = pc.participationID
18  JOIN category cat
19    ON pc.categoryID = cat.categoryID
20  GROUP BY a.archerID, c.competitionID
21  HAVING COUNT(DISTINCT pc.categoryID) > 1
22  ORDER BY categoryCount DESC, a.archerLastName;
23
```

---

#### Query 12: Generate Competition Statistics Summary (Kaung Htet Nyein)

**Purpose:** Create a comprehensive statistical summary for each competition including participation rates, average scores, and completion status. Useful for competition evaluation and reporting.

#### Why it's needed:

- Provides overview metrics for competition organizers
- Helps evaluate competition success and participation
- Supports data-driven decision making for future events
- Generates reports for stakeholders and sponsors

```
1  SELECT
2      c.competitionID,
3      c.competitionTitle,
4      c.competitionCity,
5      c.competitionStartDate,
6      c.competitionEndDate,
7      c.competitionStatus,
8      COUNT(DISTINCT p.archerID) AS totalArchers,
9      COUNT(DISTINCT r.roundID) AS totalRounds,
10     COUNT(DISTINCT rs.roundScoreID) AS totalScoresRecorded,
11     AVG(rs.totalScore) AS avgScore,
12     MAX(rs.totalScore) AS highestScore,
13     MIN(rs.totalScore) AS lowestScore,
14     SUM(rs.totalX) AS totalXsScored,
15     SUM(rs.totalTen) AS totalTensScored,
```

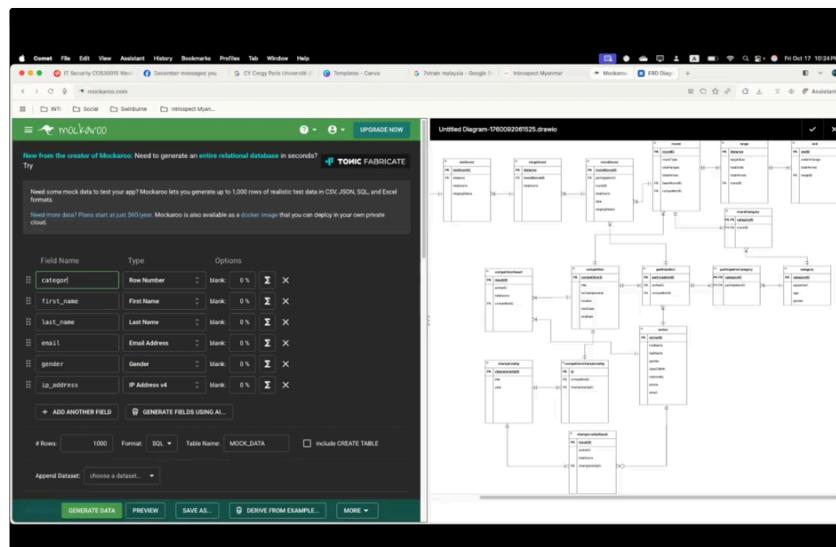
```
16   COUNT(DISTINCT cat.categoryID) AS categoriesOffered
17   FROM competition c
18   LEFT JOIN participation p ON c.competitionID = p.competitionID
19   LEFT JOIN round r ON c.competitionID = r.competitionID
20   LEFT JOIN roundScore rs ON r.roundID = rs.roundID
21   LEFT JOIN participationCategory pc ON p.participationID =
pc.participationID
22   LEFT JOIN category cat ON pc.categoryID = cat.categoryID
23   GROUP BY c.competitionID
24   ORDER BY c.competitionStartDate DESC;
25
```

---

## Table Creation

This page documents screenshots from the 17 October 2025 meeting, highlighting key steps and results as our team generated, tested, and validated mock data. Mockaroo was used to create sample datasets according to our ERD tables and field definitions. After importing data into XMAPP for local testing, successful integration will inform our migration to production.

Setting up table fields in Mockaroo to match our ERD structure.

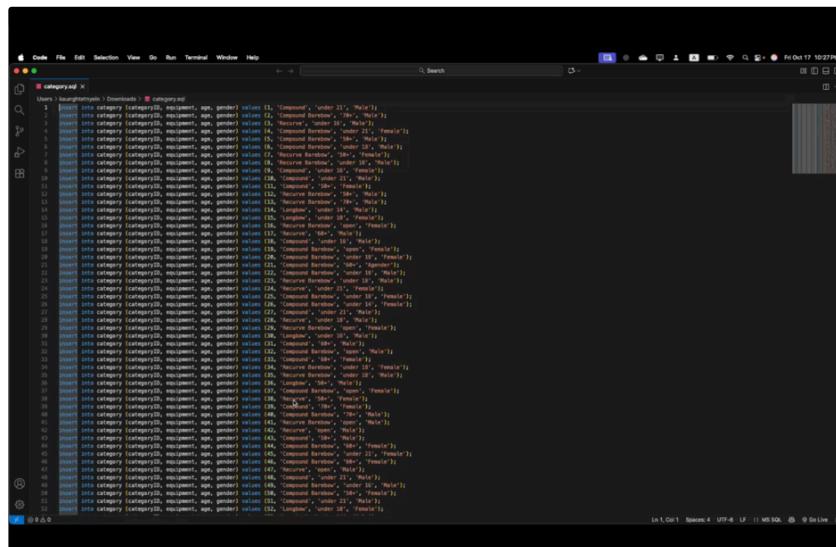


The screenshot shows the Mockaroo web application. On the left, a table lists field definitions:

| Field Name | Type          | Options |
|------------|---------------|---------|
| category   | Row Number    | Mark 0% |
| First_name | First Name    | Mark 0% |
| Last_name  | Last Name     | Mark 0% |
| Email      | Email Address | Mark 0% |
| Gender     | Gender        | Mark 0% |
| IP_Address | IP Address v4 | Mark 0% |

On the right, an Entity-Relationship (ER) diagram titled "Untitled Diagram-1f600922d6f525.drawio" illustrates the database schema. It features several entities such as "arrows", "banners", "categories", "equipment", "locations", "members", "memberships", "people", "resources", and "users". Relationships between these entities are shown as lines connecting their respective boxes.

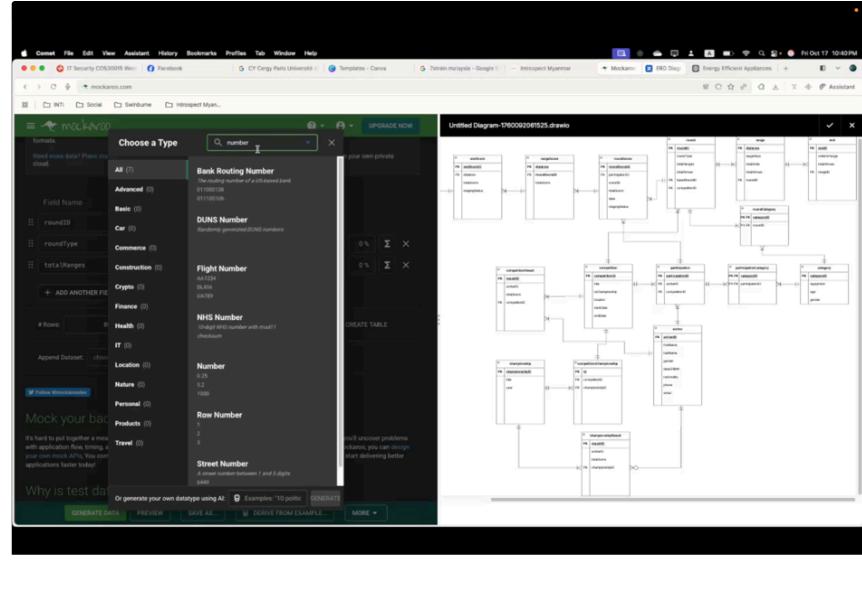
Reviewing generated sample data



```
Code : Selection : View : Go : Run : Terminal : Window : Help : Search : Fri Oct 17 10:27PM : L 1, Col 1 : Spec 4 : UTF-8 : MS SQL : Go Live : C

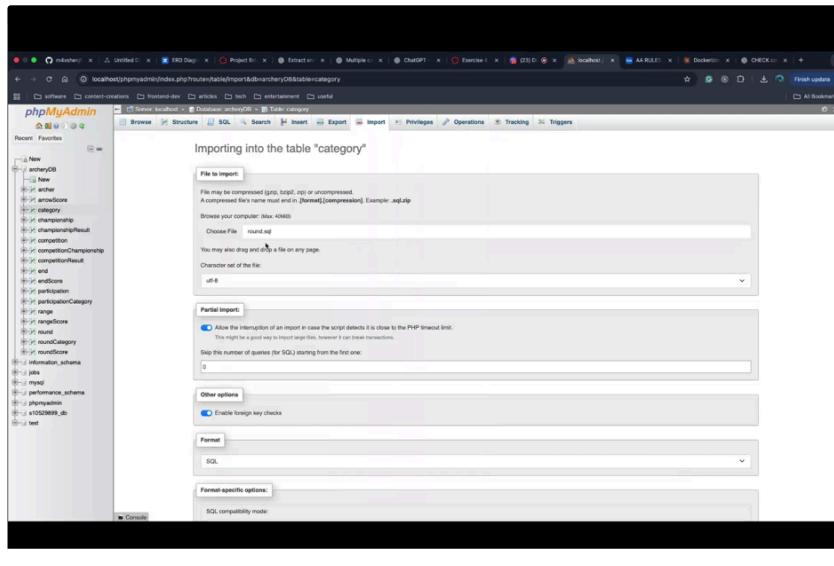
category.sql
Users : bengtmyhrm : Downloads : category.sql
1: -- Insert into category (category_id, equipment, age, gender) values (1, 'Compound', 'under 21', 'Male');
2: -- Insert into category (category_id, equipment, age, gender) values (2, 'Recurve', 'under 18', 'Male');
3: -- Insert into category (category_id, equipment, age, gender) values (3, 'Compound Barebow', '18+', 'Male');
4: -- Insert into category (category_id, equipment, age, gender) values (4, 'Recurve Barebow', '18+', 'Male');
5: -- Insert into category (category_id, equipment, age, gender) values (5, 'Compound', 'under 21', 'Female');
6: -- Insert into category (category_id, equipment, age, gender) values (6, 'Recurve', 'under 18', 'Female');
7: -- Insert into category (category_id, equipment, age, gender) values (7, 'Compound Barebow', '18+', 'Female');
8: -- Insert into category (category_id, equipment, age, gender) values (8, 'Recurve Barebow', '18+', 'Female');
9: -- Insert into category (category_id, equipment, age, gender) values (9, 'Compound', 'under 21', 'Male');
10: -- Insert into category (category_id, equipment, age, gender) values (10, 'Recurve', 'under 18', 'Male');
11: -- Insert into category (category_id, equipment, age, gender) values (11, 'Compound Barebow', '18+', 'Male');
12: -- Insert into category (category_id, equipment, age, gender) values (12, 'Recurve Barebow', '18+', 'Male');
13: -- Insert into category (category_id, equipment, age, gender) values (13, 'Compound', 'under 21', 'Female');
14: -- Insert into category (category_id, equipment, age, gender) values (14, 'Recurve', 'under 18', 'Female');
15: -- Insert into category (category_id, equipment, age, gender) values (15, 'Compound Barebow', '18+', 'Female');
16: -- Insert into category (category_id, equipment, age, gender) values (16, 'Recurve Barebow', '18+', 'Female');
17: -- Insert into category (category_id, equipment, age, gender) values (17, 'Compound', 'under 21', 'Male');
18: -- Insert into category (category_id, equipment, age, gender) values (18, 'Recurve', 'under 18', 'Male');
19: -- Insert into category (category_id, equipment, age, gender) values (19, 'Compound Barebow', '18+', 'Male');
20: -- Insert into category (category_id, equipment, age, gender) values (20, 'Recurve Barebow', '18+', 'Male');
21: -- Insert into category (category_id, equipment, age, gender) values (21, 'Compound Barebow', '18+', 'Female');
22: -- Insert into category (category_id, equipment, age, gender) values (22, 'Recurve Barebow', '18+', 'Female');
23: -- Insert into category (category_id, equipment, age, gender) values (23, 'Compound', 'under 21', 'Male');
24: -- Insert into category (category_id, equipment, age, gender) values (24, 'Recurve', 'under 18', 'Male');
25: -- Insert into category (category_id, equipment, age, gender) values (25, 'Compound Barebow', '18+', 'Male');
26: -- Insert into category (category_id, equipment, age, gender) values (26, 'Recurve Barebow', '18+', 'Male');
27: -- Insert into category (category_id, equipment, age, gender) values (27, 'Compound', 'under 21', 'Female');
28: -- Insert into category (category_id, equipment, age, gender) values (28, 'Recurve', 'under 18', 'Female');
29: -- Insert into category (category_id, equipment, age, gender) values (29, 'Compound Barebow', '18+', 'Female');
30: -- Insert into category (category_id, equipment, age, gender) values (30, 'Recurve Barebow', '18+', 'Female');
31: -- Insert into category (category_id, equipment, age, gender) values (31, 'Compound', 'under 21', 'Male');
32: -- Insert into category (category_id, equipment, age, gender) values (32, 'Recurve', 'under 18', 'Male');
33: -- Insert into category (category_id, equipment, age, gender) values (33, 'Compound Barebow', '18+', 'Male');
34: -- Insert into category (category_id, equipment, age, gender) values (34, 'Recurve Barebow', '18+', 'Male');
35: -- Insert into category (category_id, equipment, age, gender) values (35, 'Compound', 'under 21', 'Female');
36: -- Insert into category (category_id, equipment, age, gender) values (36, 'Recurve', 'under 18', 'Female');
37: -- Insert into category (category_id, equipment, age, gender) values (37, 'Compound Barebow', '18+', 'Female');
38: -- Insert into category (category_id, equipment, age, gender) values (38, 'Recurve Barebow', '18+', 'Female');
39: -- Insert into category (category_id, equipment, age, gender) values (39, 'Compound', 'under 21', 'Male');
40: -- Insert into category (category_id, equipment, age, gender) values (40, 'Recurve', 'under 18', 'Male');
41: -- Insert into category (category_id, equipment, age, gender) values (41, 'Compound Barebow', '18+', 'Male');
42: -- Insert into category (category_id, equipment, age, gender) values (42, 'Recurve Barebow', '18+', 'Male');
43: -- Insert into category (category_id, equipment, age, gender) values (43, 'Compound', 'under 21', 'Female');
44: -- Insert into category (category_id, equipment, age, gender) values (44, 'Recurve', 'under 18', 'Female');
45: -- Insert into category (category_id, equipment, age, gender) values (45, 'Compound Barebow', '18+', 'Female');
46: -- Insert into category (category_id, equipment, age, gender) values (46, 'Recurve Barebow', '18+', 'Female');
47: -- Insert into category (category_id, equipment, age, gender) values (47, 'Compound', 'under 21', 'Male');
48: -- Insert into category (category_id, equipment, age, gender) values (48, 'Recurve', 'under 18', 'Male');
49: -- Insert into category (category_id, equipment, age, gender) values (49, 'Compound Barebow', '18+', 'Male');
50: -- Insert into category (category_id, equipment, age, gender) values (50, 'Recurve Barebow', '18+', 'Male');
51: -- Insert into category (category_id, equipment, age, gender) values (51, 'Compound', 'under 21', 'Female');
52: -- Insert into category (category_id, equipment, age, gender) values (52, 'Recurve', 'under 18', 'Female');
53: -- Insert into category (category_id, equipment, age, gender) values (53, 'Compound Barebow', '18+', 'Female');
54: -- Insert into category (category_id, equipment, age, gender) values (54, 'Recurve Barebow', '18+', 'Female');
```

Creation custom list in mockaroo to add filed which is "equipment" so we can try data input like Recurve , Compound , Recurve , Longbow , Recurve Barebow .



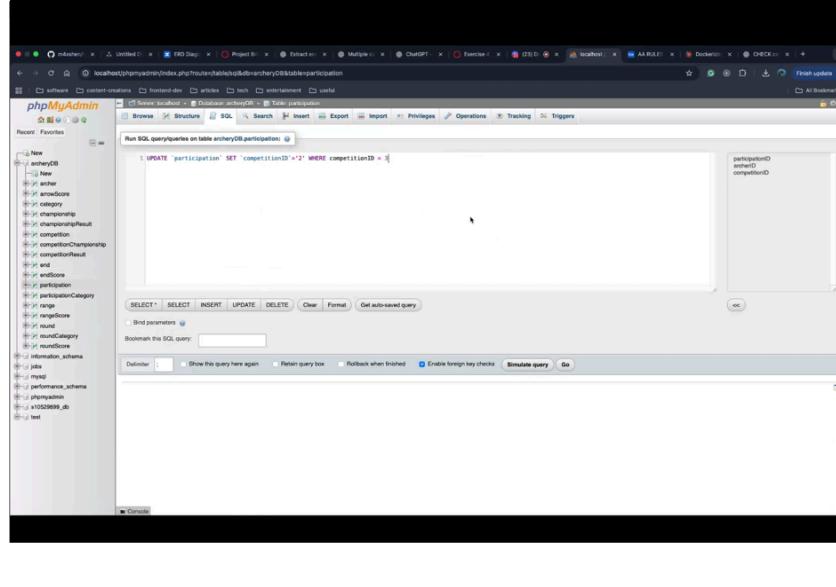
## Data import to XMAPP

Uploading CSV mock data into our local database environment.



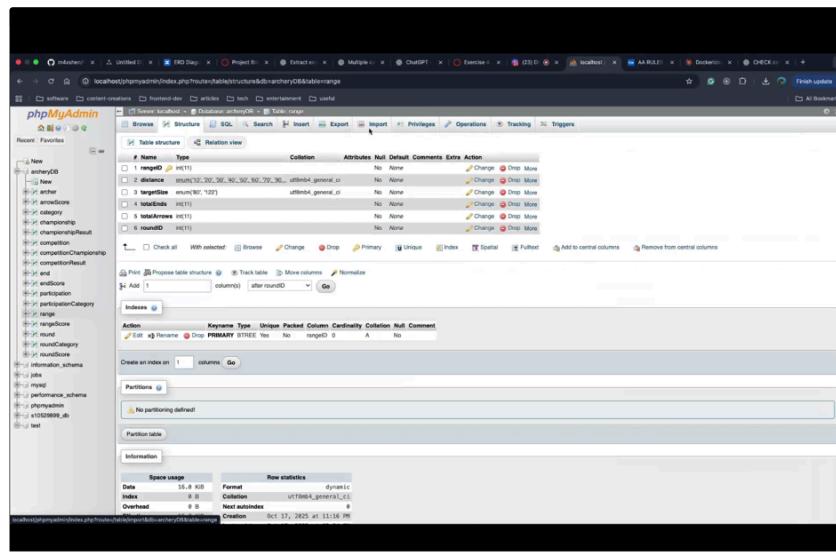
## Local table verification

Inspecting imported tables and fields for alignment with ERD.



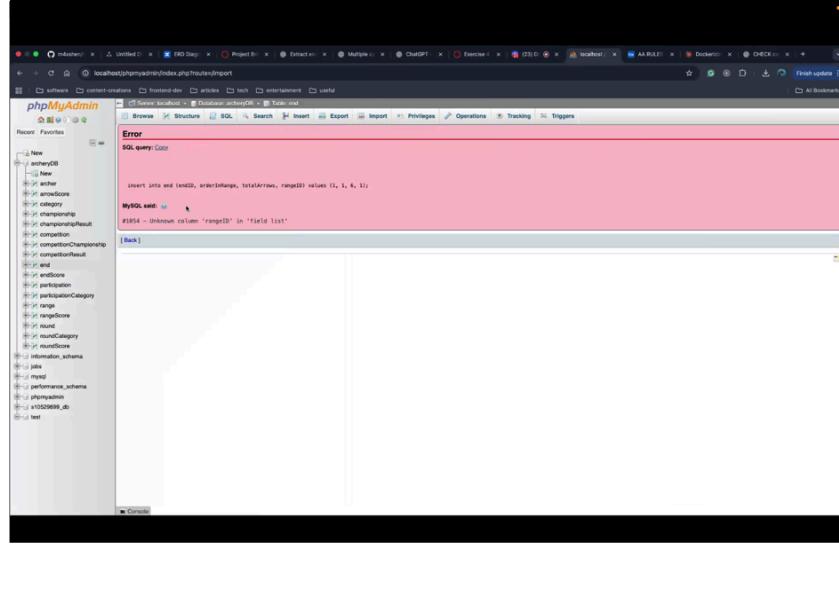
Successful import

Confirmation of data upload with no errors for this stage.



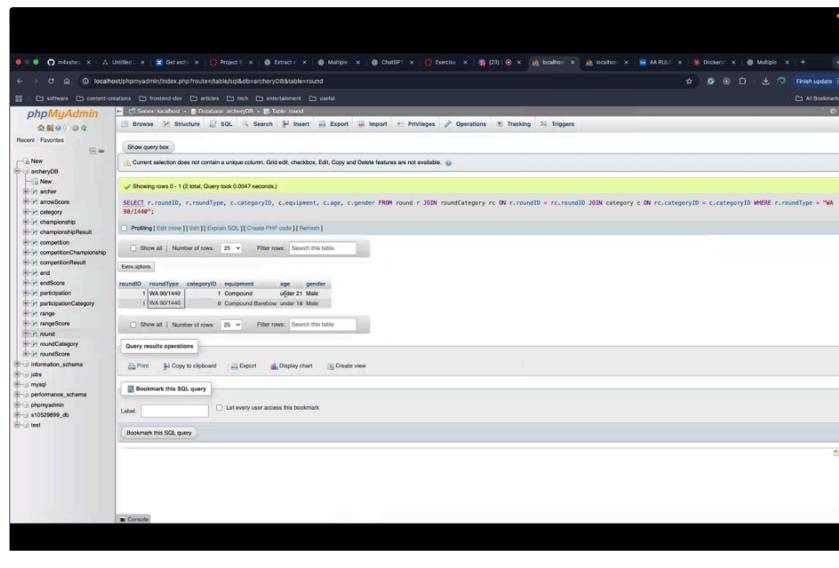
Test query results

For this one, we have solve datatype mismatch between mockaroo and sql server.



## Test query results

Running sample queries to verify field mapping and data integrity



For this SQL query is asking the database:

"Show me the name, category details, and competition title for a specific archer (whose ID is provided by the @archerID variable) who participated in the competition with ID number 1."

The screenshot shows the phpMyAdmin interface with the following details:

- Database:** archeryDB
- Table:** participation
- Rows:** 0 (Showing rows 0-4 (Total: 4))
- Time:** 0.010 seconds
- SQL Query:**

```
SELECT `archerID`, `firstName`, `lastName`, `competitionTitle`, `equipment`, `age`, `gender` FROM participation p JOIN participationCategory pc ON p.participationID = pc.participationID JOIN category cat ON pc.categoryID = cat.categoryID JOIN competition comp ON p.competitionID = comp.competitionID JOIN archer a ON p.archerID = a.archerID;
```
- Operations:** Show query box, Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.
- Table Structure:** Shows columns: archerID, firstName, lastName, competitionTitle, equipment, age, gender.
- Data:**

| archerID | firstName | lastName | competitionTitle    | equipment       | age      | gender |
|----------|-----------|----------|---------------------|-----------------|----------|--------|
| 1        | Carla     | Tabo     | Edgeware            | Compound        | under 21 | Male   |
| 2        | Edith     | Tabo     | Championship Bowbow | Compound        | over 50  | Female |
| 5        | Jacquelyn | Castillo | Edgeware            | Compound Bowbow | 50+      | Male   |
| 2        | Aquiles   | Ogibsey  | Suberbisal          | Compound Bowbow | 70+      | Male   |
| 3        | Karen     | Lutge    | Suberbisal          | Recurve         | under 16 | Male   |
- Buttons:** Show all, Number of rows, Filter rows, Search this table, Sort by key, None.
- Query results operations:** Post, Copy to clipboard, Export, Display chart, Create view.
- Bookmark this SQL query:** Label: (empty), Let every user access this bookmark.
- Console:** (empty)

Since we are testing in local environment went well, we started to create database in feenix server and created 13 tables which are listed as below:

1. archer
  2. arrowStaging
  3. category
  4. championship
  5. competition
  6. participation
  7. participationCategory
  8. range
  9. round
  10. roundType
  11. roundCategory
  12. championshipResult

### 13. roundScore

The screenshot shows the phpMyAdmin interface with the 'roundScore' table selected. The table has three columns: roundID, competitionID, and roundType. The data consists of 27 rows, each representing a different round type and its details.

| roundID | competitionID | roundType      | roundDate  |
|---------|---------------|----------------|------------|
| 4       | 40            | WA70/72        | 2024-03-15 |
| 5       | 40            | WA70/72        | 2024-03-16 |
| 6       | 40            | WA70/72        | 2024-03-17 |
| 7       | 21            | WA50/140       | 2024-06-20 |
| 8       | 21            | WA50/140       | 2024-06-22 |
| 9       | 21            | WA50/140       | 2024-06-23 |
| 10      | 22            | Long Sydney    | 2024-09-10 |
| 11      | 22            | Sydney         | 2024-09-11 |
| 12      | 22            | Brisbane       | 2024-09-12 |
| 13      | 23            | AA50/70        | 2024-12-05 |
| 14      | 23            | AA50/70        | 2024-12-06 |
| 15      | 24            | WA50/140       | 2024-04-18 |
| 16      | 24            | WA50/140       | 2024-04-19 |
| 17      | 24            | WA50/140       | 2024-04-21 |
| 18      | 25            | WA70/72        | 2024-07-08 |
| 19      | 25            | WA70/72        | 2024-07-09 |
| 20      | 25            | AA50/70        | 2024-09-11 |
| 21      | 26            | Adelaide       | 2024-05-22 |
| 22      | 26            | Short Adelaide | 2024-05-23 |
| 23      | 26            | Brisbane       | 2024-05-24 |
| 24      | 27            | Junior Cambara | 2024-08-14 |

## Data Dictionary

**Competition Table** - Stores information about each competition, like its title, dates, and location.

**Championship Table** - Contains each championship's basic details, such as name and year.

**Round Table** - Represents each round within a competition, including its format and date.

**RoundType Table** - Defines different formats (types) for rounds (e.g., WA70, Indoor18).

**Range Table** - Details the distances, target sizes, and arrow arrangement for rounds.

**Archer Table** - Holds personal info for each archer: name, gender, contact details, etc.

**Category Table** - Describes age group, gender, and equipment type categories for archers.

**RoundCategory Table** - Links rounds to the categories that participate in them.

**ParticipationCategory Table** - Connects participation records to specific categories.

**Participation Table** - Tracks each archer's entry into a competition.

**ArrowStaging Table** - Records scores for each arrow shot, including confirmation status and inner 10 marks.

**RoundScore Table** - Summarizes total scores and key stats (like X and 10s) for a round per participant.

**ChampionshipResult Table** - Displays final results for each archer in a championship, including overall scores and categories.

## Testing Query

Query a List of Competition Summary

Showing rows 0 - 24 (0 total). Query took 0.0020 seconds. (1 query(s) run)

**Structure** **SQL** **Search** **Query** **Export** **Import** **Operations** **Routines** **Tracking** **Designer** **Central columns**

Current selection does not contain a unique column. Grid edit, reindex, Edit, Copy and Delete features are not available.

Archer

View definition

ALTER VIEW Archer AS SELECT competitionID, competitionTitle, competitionStartDate, competitionEndDate, competitionLocation, roundID, roundDate, roundType, baseRoundID, numberOftargets, distances, totalArrowheadsRound FROM competition WHERE competitionID IN (SELECT competitionID FROM competition WHERE competitionTitle = 'National Archery Series' AND competitionLocation = 'Australia') AND competitionID NOT IN (SELECT competitionID FROM competition WHERE competitionTitle = 'World Archery Series' AND competitionLocation = 'Australia')

Extra options

competitionID competitionTitle competitionStartDate competitionEndDate competitionLocation roundID roundDate roundType baseRoundID numberOftargets distances totalArrowheadsRound

|    |                                    |            |            |                                     |    |            |                    |             |                    |                    |     |
|----|------------------------------------|------------|------------|-------------------------------------|----|------------|--------------------|-------------|--------------------|--------------------|-----|
| 21 | Spring Classic Indoor Championship | 2024-03-15 | 2024-03-17 | National Archery Center, London     | 7  | 2024-03-09 | WAM001440          | 4           | 90m, 70m, 50m, 30m | 144                |     |
| 21 | Spring Classic Indoor Championship | 2024-03-15 | 2024-03-17 | National Archery Center, London     | 8  | 2024-03-22 | WAT001440          | 4           | 70m, 50m, 30m      | 144                |     |
| 21 | Spring Classic Indoor Championship | 2024-03-15 | 2024-03-17 | National Archery Center, London     | 9  | 2024-03-29 | WAM001440          | 4           | 90m, 70m, 50m, 30m | 144                |     |
| 21 | Spring Classic Indoor Championship | 2024-03-15 | 2024-03-17 | National Archery Center, London     | 10 | 2024-04-10 | Long Sydney        | N/A         | 240                |                    |     |
| 22 | Summer Outdoor Tournament          | 2024-06-20 | 2024-06-23 | Olympic Archery Range, Paris        | 11 | 2024-06-11 | Long Sydney        | 4           | 70m, 50m, 30m, 40m | 240                |     |
| 22 | Summer Outdoor Tournament          | 2024-06-20 | 2024-06-23 | Olympic Archery Range, Paris        | 12 | 2024-06-18 | Long Sydney        | 4           | 90m, 70m, 50m, 40m | 240                |     |
| 22 | Summer Outdoor Tournament          | 2024-06-20 | 2024-06-23 | Olympic Archery Range, Paris        | 13 | 2024-06-25 | Wolongong          | 1           | 30m                | 144                |     |
| 22 | Autumn Cup Qualifier               | 2024-09-10 | 2024-09-12 | Regional Sports Complex, Manchester | 14 | 2024-09-10 | AAS00700           | Wolongong   | 1                  | 20m                | 144 |
| 22 | Autumn Cup Qualifier               | 2024-09-10 | 2024-09-12 | Regional Sports Complex, Manchester | 15 | 2024-09-17 | Wolongong          | 4           | 90m, 70m, 50m, 30m | 144                |     |
| 22 | Autumn Cup Qualifier               | 2024-09-10 | 2024-09-12 | Regional Sports Complex, Manchester | 16 | 2024-09-24 | WAM001440          | 4           | 70m, 50m, 30m      | 144                |     |
| 24 | Winter Indoor Series - Round 1     | 2024-12-05 | 2024-12-06 | Indoor Arena, Berlin                | 17 | 2024-12-01 | WAM001440          | 4           | 60m, 50m, 40m, 30m | 144                |     |
| 24 | Winter Indoor Series - Round 1     | 2024-12-05 | 2024-12-06 | Indoor Arena, Berlin                | 18 | 2024-12-08 | Wolongong          | 1           | 30m                | 144                |     |
| 25 | National Target Championships      | 2024-04-18 | 2024-04-21 | State Archery Grounds, Sydney       | 19 | 2024-04-10 | WAM00700           | Wolongong   | 1                  | 60m                | 144 |
| 25 | National Target Championships      | 2024-04-18 | 2024-04-21 | State Archery Grounds, Sydney       | 20 | 2024-04-11 | AAS00700           | Wolongong   | 1                  | 60m                | 144 |
| 25 | National Target Championships      | 2024-04-18 | 2024-04-21 | State Archery Grounds, Sydney       | 21 | 2024-04-18 | Long Sydney        | 4           | 90m, 70m, 50m, 30m | 144                |     |
| 26 | International Open Competition     | 2024-07-08 | 2024-07-11 | World Archery Center, Seoul         | 22 | 2024-07-23 | Short Adelaide     | Long Sydney | 4                  | 50m, 40m, 30m, 20m | 240 |
| 26 | International Open Competition     | 2024-07-08 | 2024-07-11 | World Archery Center, Seoul         | 23 | 2024-07-23 | Brisbane           | Long Sydney | 4                  | 70m, 50m, 30m, 40m | 240 |
| 27 | Regional Bowline Challenge         | 2024-05-27 | 2024-05-28 | County Sports Park, Dublin          | 24 | 2024-05-24 | Wolongong          | 3           | 30m, 20m, 10m      | 180                |     |
| 27 | Regional Bowline Challenge         | 2024-05-27 | 2024-05-28 | County Sports Park, Dublin          | 25 | 2024-05-25 | Minicampers Hobart | 3           | 30m, 20m, 10m      | 180                |     |
| 28 | Youth Development Tournament       | 2024-08-14 | 2024-08-16 | Junior Training Center, Toronto     | 26 | 2024-08-16 | AAS00700           | Wolongong   | 1                  | 40m                | 144 |
| 28 | Youth Development Tournament       | 2024-08-14 | 2024-08-16 | Junior Training Center, Toronto     | 27 | 2024-08-17 | Melbourne          | 4           | 90m, 70m, 50m      | 144                |     |
| 28 | Youth Development Tournament       | 2024-08-14 | 2024-08-16 | Junior Training Center, Toronto     | 28 | 2024-08-18 | Pert               | Hobart      | 3                  | 70m, 60m, 50m      | 144 |

## Query a List of Archer

Showing rows 0 - 22 (22 total). Query took 0.0020 seconds.

**Structure** **SQL** **Search** **Query** **Export** **Import** **Operations** **Tracking**

Polite! Edit view | Log | Export SQL | Create PHP code | Refresh

Show all | Number of rows: 25 | Filter rows | Search this table | Sort by key: None

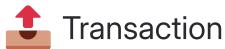
Extra options

archer

| archerFirstName | archerLastName | archerNationality |
|-----------------|----------------|-------------------|
| Campbell        | Zoe            | New Zealand       |
| Roberts         | Michael        | New Zealand       |
| Phillips        | Christopher    | New Zealand       |
| Evens           | Isaac          | New Zealand       |
| Turner          | Hannah         | New Zealand       |
| Somes           | Gabriel        | New Zealand       |
| Parker          | Olivia         | New Zealand       |
| Collier         | Anthony        | New Zealand       |
| Green           | Patsy          | New Zealand       |
| Jones           | Andrew         | New Zealand       |
| Reyes           | Clare          | New Zealand       |
| Cruz            | Christopher    | New Zealand       |
| Hughes          | Samuel         | New Zealand       |
| Pringle         | Isaac          | New Zealand       |
| Myles           | Amelia         | New Zealand       |
| Long            | Liam           | New Zealand       |
| Foster          | Caroline       | New Zealand       |
| Samuels         | Jayden         | New Zealand       |
| Bulter          | Styler         | New Zealand       |

## SQL Final File

```
cos20031.sql
17 Nov 2025, 03:02 PM
```



## Transaction

# Archery Competition Database - Transaction Scenarios

## Overview

This document provides real-world transaction scenarios for the archery competition management system. Each transaction includes a title, justification, and complete SQL with BEGIN/COMMIT statements.

---

## 1. ARCHER MANAGEMENT TRANSACTIONS

### Transaction 1.1: New Archer Registration

#### Reasons Why Needed:

- New users need to create accounts to participate in competitions
- Essential for user onboarding and authentication
- Foundation for all archer-related activities in the system

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 INSERT INTO archer (firstName, lastName, gender, dateOfBirth,
4 nationality, phone, email, password)
5 VALUES ('John', 'Smith', 'M', '1995-06-15', 'USA', '+1234567890',
6 'john.smith@email.com', '$2y$10$hashedpassword');
7 COMMIT;
```

---

### Transaction 1.2: Update Archer Profile Information

#### Reasons Why Needed:

- Archers need to update contact information (phone, email)
- Personal information may change over time
- Important for communication and notifications

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 UPDATE archer
4 SET phone = '+1987654321',
5     email = 'john.newemail@email.com'
6 WHERE archerID = 1;
7
8 COMMIT;
9
```

---

### Transaction 1.3: Change Archer Password

#### Reasons Why Needed:

- Security requirement for password resets
- Users may forget passwords or want to update them
- Essential for account security

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 UPDATE archer
4 SET password = '$2y$10$newhashedpassword'
5 WHERE archerID = 1 AND email = 'john.smith@email.com';
6
7 COMMIT;
8
```

---

#### Transaction 1.4: Delete Archer Account

#### Reasons Why NOT Needed:

- Rarely needed as historical data should be preserved
- Violates data integrity if archer has competition records
- Better to use soft delete or archive status instead
- **Recommendation:** Implement account deactivation rather than deletion

#### Transaction Query (NOT RECOMMENDED):

```
1 -- NOT RECOMMENDED: Use account deactivation instead
2 BEGIN TRANSACTION;
3
4 DELETE FROM archer
5 WHERE archerID = 1;
6
7 COMMIT;
8
```

---

## 2. CHAMPIONSHIP & COMPETITION MANAGEMENT

#### Transaction 2.1: Create New Championship

#### Reasons Why Needed:

- Establish a new championship series (e.g., World Championships 2025)
- Required before creating competitions under this championship
- Done once per major tournament series

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 INSERT INTO championship (title, year)
4 VALUES ('World Archery Championships', 2025);
5
6 COMMIT;
7
```

---

#### Transaction 2.2: Create New Competition Event

#### Reasons Why Needed:

- Set up individual competition events within a championship
- Essential for organizing archery tournaments
- Required before archers can register

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 INSERT INTO competition (title, location, startDate, endDate,
4   championshipID, status)
5 VALUES (
6   'National Qualifier - Western Region',
7   'Los Angeles, CA',
8   '2025-03-15',
9   '2025-03-17',
10  1,
11  'scheduled'
12 );
13
14 COMMIT;
```

---

#### Transaction 2.3: Update Competition Status

##### Reasons Why Needed:

- Track competition lifecycle (scheduled → ongoing → completed)
- Essential for managing active vs completed events
- Used throughout competition day operations

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 -- Move competition from scheduled to ongoing
4 UPDATE competition
5 SET status = 'ongoing'
6 WHERE competitionID = 1 AND status = 'scheduled';
7
8 COMMIT;
```

```
1 BEGIN TRANSACTION;
2
3 -- Mark competition as completed
4 UPDATE competition
5 SET status = 'completed'
6 WHERE competitionID = 1 AND status = 'ongoing';
7
8 COMMIT;
```

---

#### Transaction 2.4: Reschedule Competition Dates

##### Reasons Why Needed:

- Weather delays or logistical issues may require rescheduling
- Important for keeping participants informed
- Updates must be coordinated with related rounds

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 UPDATE competition
4 SET startDate = '2025-03-16',
5     endDate = '2025-03-18'
6 WHERE competitionID = 1 AND status = 'scheduled';
7
8 COMMIT;
9
```

### 3. COMPETITION REGISTRATION TRANSACTIONS

#### Transaction 3.1: Archer Competition Registration (Simple)

##### Reasons Why Needed:

- Register archer for a competition
- Most common transaction during registration periods
- Foundation for all scoring and participation tracking

##### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 INSERT INTO participation (archerID, competitionID)
4 VALUES (1, 5);
5
6 COMMIT;
7
```

#### Transaction 3.2: Archer Competition Registration with Category Assignment

##### Reasons Why Needed:

- Complete registration process requires category assignment
- Determines which division archer competes in
- Ensures proper categorization for fair competition
- **Most realistic registration workflow**

##### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 -- Step 1: Create participation record
4 INSERT INTO participation (archerID, competitionID)
5 VALUES (1, 5);
6
7 -- Get the new participation ID
8 SET @participationID = LAST_INSERT_ID();
9
10 -- Step 2: Assign archer to appropriate category
11 INSERT INTO participationCategory (categoryID, participationID)
12 VALUES (3, @participationID);
13
14 COMMIT;
15
```

#### Transaction 3.3: Bulk Archer Registration

##### Reasons Why Needed:

- Register multiple archers at once (team registrations)
- Efficient for club or team sign-ups
- Reduces database round trips

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3     INSERT INTO participation (archerID, competitionID)
4     VALUES
5         (1, 5),
6         (2, 5),
7         (3, 5),
8         (4, 5),
9         (5, 5);
10
11    COMMIT;
12
```

---

#### Transaction 3.4: Withdraw Archer from Competition

##### Reasons Why Needed:

- Archer may need to cancel participation (injury, schedule conflict)
- Must be done before competition starts
- Frees up competition slot

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3     -- First delete category assignment
4     DELETE FROM participationCategory
5     WHERE participationID = (
6         SELECT participationID
7             FROM participation
8                 WHERE archerID = 1 AND competitionID = 5
9     );
10
11     -- Then delete participation
12     DELETE FROM participation
13     WHERE archerID = 1 AND competitionID = 5;
14
15     COMMIT;
16
```

---

#### Transaction 3.5: Change Archer Category After Registration

##### Reasons Why Needed:

- Archer may have registered in wrong category
- Equipment change requires category change
- Must be corrected before competition starts

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3     UPDATE participationCategory
4     SET categoryID = 5
5     WHERE participationID = 10;
6
```

```
7 COMMIT;  
8
```

## 4. COMPETITION SETUP TRANSACTIONS

### Transaction 4.1: Setup Competition Rounds

#### Reasons Why Needed:

- Define shooting rounds for a competition
- Required before any scoring can begin
- Establishes competition structure

#### Transaction Query:

```
1 BEGIN TRANSACTION;  
2  
3     -- Create multiple rounds for a 3-day competition  
4     INSERT INTO round (competitionID, date, roundType)  
5     VALUES  
6         (1, '2025-03-15', 1),    -- Day 1: Qualification Round  
7         (1, '2025-03-16', 1),    -- Day 2: Qualification Round  
8         (1, '2025-03-17', 2);   -- Day 3: Final Round  
9  
10    COMMIT;  
11
```

### Transaction 4.2: Assign Categories to Round

#### Reasons Why Needed:

- Define which categories shoot in which rounds
- Enables scheduling different divisions at different times
- Essential for large competitions with multiple categories

#### Transaction Query:

```
1 BEGIN TRANSACTION;  
2  
3     -- Assign all senior categories to round 5  
4     INSERT INTO roundCategory (categoryID, roundID)  
5     VALUES  
6         (1, 5),    -- Recurve Senior Male  
7         (2, 5),    -- Recurve Senior Female  
8         (3, 5),    -- Compound Senior Male  
9         (4, 5);   -- Compound Senior Female  
10  
11    COMMIT;  
12
```

### Transaction 4.3: Update Round Schedule

#### Reasons Why Needed:

- Weather or operational delays may require rescheduling
- Important to update before archers arrive
- Minimal transaction needed

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 UPDATE round
4 SET date = '2025-03-16'
5 WHERE roundID = 1;
6
7 COMMIT;
8
```

## 5. LIVE SCORING TRANSACTIONS

Transaction 5.1: Record Single Arrow Score

### Reasons Why Needed:

- Basic unit of scoring during competition
- Happens continuously during shooting
- **Highest frequency transaction in the system**

Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 INSERT INTO arrowStaging (
4     roundID,
5     participationID,
6     distance,
7     endOrder,
8     arrowScore,
9     stagingStatus,
10    isX,
11    date,
12    recorderID
13 )
14 VALUES (
15     5,                               -- roundID
16     10,                             -- participationID
17     70,                             -- distance in meters
18     1,                               -- end number
19     10,                             -- arrow score
20     'pending',                      -- needs verification
21     TRUE,                           -- is an X (inner 10)
22     CURRENT_TIMESTAMP,              -- when recorded
23     3                               -- recorder who entered it
24 );
25
26 COMMIT;
27
```

Transaction 5.2: Record Complete End (6 Arrows)

### Reasons Why Needed:

- More efficient than 6 individual transactions
- Common workflow: record all arrows in an end at once
- Reduces database overhead during live scoring
- **Recommended approach for live scoring**

Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 INSERT INTO arrowStaging (
```

```

4     roundID,
5     participationID,
6     distance,
7     endOrder,
8     arrowScore,
9     stagingStatus,
10    isX,
11    date,
12    recorderID
13 )
14 VALUES
15   (5, 10, 70, 1, 10, 'pending', TRUE, CURRENT_TIMESTAMP, 3),
16   (5, 10, 70, 1, 9, 'pending', FALSE, CURRENT_TIMESTAMP, 3),
17   (5, 10, 70, 1, 10, 'pending', FALSE, CURRENT_TIMESTAMP, 3),
18   (5, 10, 70, 1, 10, 'pending', TRUE, CURRENT_TIMESTAMP, 3),
19   (5, 10, 70, 1, 8, 'pending', FALSE, CURRENT_TIMESTAMP, 3),
20   (5, 10, 70, 1, 9, 'pending', FALSE, CURRENT_TIMESTAMP, 3);
21
22 COMMIT;
23

```

#### Transaction 5.3: Verify Arrow Scores

##### Reasons Why Needed:

- Scores must be verified by judges before becoming official
- Implements approval workflow for score accuracy
- Prevents score manipulation

##### Transaction Query:

```

1 BEGIN TRANSACTION;

2
3 -- Verify all arrows in a specific end
4 UPDATE arrowStaging
5 SET stagingStatus = 'verified'
6 WHERE participationID = 10
7   AND roundID = 5
8   AND endOrder = 1
9   AND stagingStatus = 'pending';
10
11 COMMIT;
12

```

#### Transaction 5.4: Correct Arrow Score

##### Reasons Why Needed:

- Score entry errors need correction
- Dispute resolution requires score adjustment
- Must be done before verification

##### Transaction Query:

```

1 BEGIN TRANSACTION;

2
3 UPDATE arrowStaging
4 SET arrowScore = 9,
5     isX = FALSE
6 WHERE arrowStagingID = 123
7   AND stagingStatus = 'pending';

8
9 COMMIT;
10

```

---

#### Transaction 5.5: Bulk Verify Round Scores

##### Reasons Why Needed:

- After end of shooting, verify all scores for a round
- Efficient batch verification process
- Prepares data for final scoring

##### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 UPDATE arrowStaging
4 SET stagingStatus = 'verified'
5 WHERE roundID = 5
6   AND stagingStatus = 'pending';
7
8 COMMIT;
9
```

---

## 6. SCORE AGGREGATION TRANSACTIONS

#### Transaction 6.1: Calculate and Insert Round Score from Arrow Data

##### Reasons Why Needed:

- Aggregate individual arrow scores into round total
- Creates official round result
- Required for leaderboards and rankings
- **Critical transaction after round completion**

##### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3 -- Calculate totals from verified arrow staging data
4 INSERT INTO roundScore (participationID, roundID, totalScore, totalX,
5   totalTen, dateRecorded)
6 SELECT
7   participationID,
8   roundID,
9   SUM(arrowScore) AS totalScore,
10  SUM(CASE WHEN isX THEN 1 ELSE 0 END) AS totalX,
11  SUM(CASE WHEN arrowScore = 10 THEN 1 ELSE 0 END) AS totalTen,
12  CURRENT_TIMESTAMP AS dateRecorded
13 FROM arrowStaging
14 WHERE participationID = 10
15   AND roundID = 5
16   AND stagingStatus = 'verified'
17 GROUP BY participationID, roundID;
18
19 COMMIT;
20
```

---

#### Transaction 6.2: Update Round Score (Correction)

##### Reasons Why Needed:

- Score recalculation may be needed after appeal
- Correction of calculation errors

- Less common but necessary for accuracy

#### **Transaction Query:**

```

1 BEGIN TRANSACTION;
2
3 UPDATE roundScore
4   SET totalScore = 660,
5       totalX = 25,
6       totalTen = 43,
7       dateRecorded = CURRENT_TIMESTAMP
8 WHERE roundScoreID = 1;
9
10 COMMIT;
11

```

#### Transaction 6.3: Batch Calculate Round Scores for Entire Round

#### **Reasons Why Needed:**

- After round completion, calculate all archer scores at once
- More efficient than individual calculations
- Ensures all scores are ready for leaderboard

#### **Transaction Query:**

```

1 BEGIN TRANSACTION;
2
3 INSERT INTO roundScore (participationID, roundID, totalScore, totalX,
4   totalTen, dateRecorded)
5 SELECT
6   participationID,
7   roundID,
8   SUM(arrowScore) as totalScore,
9   SUM(CASE WHEN isX THEN 1 ELSE 0 END) as totalX,
10  SUM(CASE WHEN arrowScore = 10 THEN 1 ELSE 0 END) as totalTen,
11  CURRENT_TIMESTAMP as dateRecorded
12 FROM arrowStaging
13 WHERE roundID = 5
14   AND stagingStatus = 'verified'
15 GROUP BY participationID, roundID;
16
17 COMMIT;
18

```

## 7. CHAMPIONSHIP RESULT TRANSACTIONS

#### Transaction 7.1: Calculate Championship Result for Single Archer

#### **Reasons Why Needed:**

- Aggregate all round scores into championship total
- Creates official championship standings
- Done after all competition rounds are complete

#### **Transaction Query:**

```

1 BEGIN TRANSACTION;
2
3 INSERT INTO championshipResult (archerID, championshipID, totalScore,
4   age, gender, equipment)
5 SELECT
6   p.archerID,
7   r.championshipID,
8   SUM(score) as totalScore,
9   p.age,
10  p.gender,
11  p.equipment
12 FROM participation p
13 JOIN roundScore r ON p.participationID = r.participationID
14 WHERE p.championshipID = 1
15 GROUP BY p.archerID, r.championshipID;
16
17 COMMIT;
18

```

```

6     comp.championshipID,
7     SUM(rs.totalScore) as totalScore,
8     CASE
9       WHEN TIMESTAMPDIFF(YEAR, a.dateOfBirth, CURRENT_DATE) < 18
10      THEN 'Junior'
11        WHEN TIMESTAMPDIFF(YEAR, a.dateOfBirth, CURRENT_DATE) < 50
12      THEN 'Senior'
13        ELSE 'Master'
14      END as age,
15     a.gender,
16     cat.equipment
17   FROM roundScore rs
18   JOIN round r ON rs.roundID = r.roundID
19   JOIN participation p ON rs.participationID = p.participationID
20   JOIN competition comp ON p.competitionID = comp.competitionID
21   JOIN archer a ON p.archerID = a.archerID
22   JOIN participationCategory pc ON p.participationID =
23     pc.participationID
24   JOIN category cat ON pc.categoryID = cat.categoryID
25   WHERE p.archerID = 1
26     AND comp.championshipID = 1
27   GROUP BY p.archerID, comp.championshipID, a.dateOfBirth, a.gender,
cat.equipment;
28
29 COMMIT;
30
31

```

---

#### Transaction 7.2: Batch Calculate Championship Results for All Archers

##### Reasons Why Needed:

- Calculate championship standings for all participants
- Done once after championship completion
- Generates final rankings

##### Transaction Query:

```

1 BEGIN TRANSACTION;
2
3 INSERT INTO championshipResult (archerID, championshipID, totalScore,
4   age, gender, equipment)
5   SELECT
6     p.archerID,
7     comp.championshipID,
8     SUM(rs.totalScore) as totalScore,
9     CASE
10       WHEN TIMESTAMPDIFF(YEAR, a.dateOfBirth, comp.startDate) < 18
11      THEN 'Junior'
12        WHEN TIMESTAMPDIFF(YEAR, a.dateOfBirth, comp.startDate) < 50
13      THEN 'Senior'
14        ELSE 'Master'
15      END as age,
16     a.gender,
17     cat.equipment
18   FROM roundScore rs
19   JOIN round r ON rs.roundID = r.roundID
20   JOIN participation p ON rs.participationID = p.participationID
21   JOIN competition comp ON p.competitionID = comp.competitionID
22   JOIN archer a ON p.archerID = a.archerID
23   JOIN participationCategory pc ON p.participationID =
24     pc.participationID
25   JOIN category cat ON pc.categoryID = cat.categoryID
26   WHERE comp.championshipID = 1
27   GROUP BY p.archerID, comp.championshipID, a.dateOfBirth,
comp.startDate, a.gender, cat.equipment;
28
29 COMMIT;
30
31

```

### Transaction 7.3: Update Championship Result

#### Reasons Why Needed:

- Correction after appeal or recalculation
- Update after late score adjustments
- Rare but necessary for accuracy

#### Transaction Query:

```

1 BEGIN TRANSACTION;
2
3 UPDATE championshipResult
4 SET totalScore = 1335
5 WHERE archerID = 1 AND championshipID = 1;
6
7 COMMIT;
8

```

## 8. RECORDER MANAGEMENT TRANSACTIONS

### Transaction 8.1: Register New Score Recorder

#### Reasons Why Needed:

- Add judges/officials who can enter scores
- Required for score entry accountability
- Done during competition preparation

#### Transaction Query:

```

1 BEGIN TRANSACTION;
2
3 INSERT INTO recorder (firstName, lastName, email, password)
4 VALUES ('Jane', 'Doe', 'jane.doe@archery.org',
5 '$2y$10$hashedpassword');
6
7 COMMIT;
8

```

### Transaction 8.2: Update Recorder Credentials

#### Reasons Why Needed:

- Recorder may need to update email or reset password
- Maintain current contact information
- Security updates

#### Transaction Query:

```

1 BEGIN TRANSACTION;
2
3 UPDATE recorder
4 SET email = 'jane.newemail@archery.org',
5     password = '$2y$10$newhashedpassword'
6 WHERE recorderID = 1;
7
8 COMMIT;
9

```

## 9. REFERENCE DATA SETUP TRANSACTIONS

Transaction 9.1: Initialize Category Reference Data

**Reasons Why NOT Frequently Needed:**

- Done once during system initialization
- Categories are predefined by archery regulations
- Rarely changes (only when new equipment types introduced)
- **Not a regular operational transaction**

**Transaction Query:**

```

1 BEGIN TRANSACTION;
2
3 INSERT INTO category (equipment, age, gender)
4 VALUES
5     ('Recurve', 'Senior', 'M'),
6     ('Recurve', 'Senior', 'F'),
7     ('Compound', 'Senior', 'M'),
8     ('Compound', 'Senior', 'F'),
9     ('Recurve', 'Junior', 'M'),
10    ('Recurve', 'Junior', 'F'),
11    ('Compound', 'Junior', 'M'),
12    ('Compound', 'Junior', 'F'),
13    ('Barebow', 'Senior', 'M'),
14    ('Barebow', 'Senior', 'F');
15
16 COMMIT;
17

```

Transaction 9.2: Initialize Round Types

**Reasons Why NOT Frequently Needed:**

- Setup transaction done once
- Defines standard archery round formats (WA 70m, FITA, etc.)
- Based on World Archery regulations
- **Not a regular operational transaction**

**Transaction Query:**

```

1 BEGIN TRANSACTION;
2
3 INSERT INTO roundType (roundType, baseRoundTypeID)
4 VALUES
5     ('WA 70m Round', NULL),
6     ('WA 1440 Round', NULL),
7     ('FITA Round', NULL),
8     ('WA 720 Round', NULL),
9     ('Indoor 18m Round', NULL);
10
11 COMMIT;
12

```

Transaction 9.3: Initialize Range Configurations

**Reasons Why NOT Frequently Needed:**

- Setup transaction for standard range configurations
- Defines distance, target size, arrows per end
- Based on archery regulations
- **Not a regular operational transaction**

#### Transaction Query:

```
1 BEGIN TRANSACTION;
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
```