Create Tables Script

```sql
create table Championship
(
    ChampionshipID smallint unsigned auto_increment,
    ChampionshipName varchar(100),
    ChampionshipYear year unique,
    primary key (ChampionshipID)
);

create table Competition
(
    CompetitionID smallint unsigned auto_increment,
    ChampionshipID smallint unsigned,
    CompetitionName varchar(100),
    CompetitionDate date,
    primary key (CompetitionID),
    foreign key (ChampionshipID) references Championship(ChampionshipID)
);

create table Round
(
    RoundCode varchar(20),
    primary key (RoundCode)
);

create table `Range`
(
    RangeID int unsigned auto_increment,
    RoundCode varchar(20),
    Distance enum('90', '70', '60', '50', '40', '30', '20', '10'),
    EndCount enum('5', '6'),
    FaceSize enum('80', '122'),
    primary key (RangeID),
    foreign key (RoundCode) references Round(RoundCode)
);

create table EquivalentRound
(
    RoundCode varchar(20),
    EquivalentRoundCode varchar(20),
    Valid bit(1),
    primary key (RoundCode, EquivalentRoundCode),
    foreign key (RoundCode) references Round(RoundCode)
        on update cascade,
    foreign key (EquivalentRoundCode) references Round(RoundCode)
        on update cascade
);

create table Archer
(
    ArcherID int unsigned auto_increment,
    FirstName varchar(50),
    LastName varchar(50),
    ArcherAge tinyint unsigned,
    Gender enum('M', 'F'),
    primary key (ArcherID)
);

create table Class
(
    ClassName varchar(100),
    AgeLimitMin tinyint unsigned,
    AgeLimitMax tinyint unsigned,
    Gender enum('M', 'F'),
    primary key (ClassName)
);

create table Category
(
    CategoryName varchar(100),
    ClassName varchar(100),
    Division enum('Recurve', 'Barebow', 'Longbow', 'Compound'),
    primary key (CategoryName),
    foreign key (ClassName) references Class(ClassName)
);

create table RoundCategory
(
    RoundCode varchar(20),
    CategoryName varchar(100),
    primary key (RoundCode, CategoryName),
    foreign key (RoundCode) references Round(RoundCode)
        on update cascade,
    foreign key (CategoryName) references Category(CategoryName)
        on update cascade
);

create table RoundResult
(
    RoundResultID int unsigned auto_increment,
    ArcherID int unsigned,
    RoundCode varchar(20),
    CategoryName varchar(100),
    CompetitionID smallint unsigned default null,
    Result int unsigned,
    ResultDate date,
    primary key (RoundResultID),
    foreign key (ArcherID) references Archer(ArcherID),
    foreign key (RoundCode) references Round(RoundCode)
        on update cascade,
    foreign key (CategoryName) references Category(CategoryName)
        on update cascade,
    foreign key (CompetitionID) references Competition(CompetitionID)
);

create table Score
(
    ScoreID int unsigned auto_increment,
    RoundResultID int unsigned,
    RangeIndex tinyint unsigned,
    EndIndex tinyint unsigned,
    Arrow1 tinyint unsigned,
    Arrow2 tinyint unsigned,
    Arrow3 tinyint unsigned,
    Arrow4 tinyint unsigned,
    Arrow5 tinyint unsigned,
    Arrow6 tinyint unsigned,
    primary key (ScoreID),
    foreign key (RoundResultID) references RoundResult(RoundResultID)
);

```

Table Documentation

This page is used to document every entity in the archery database.

Championship

This table stores the annual club championship.

Championship		
Field Name	**Data Type**	**Field Purpose**
ChampionshipID (PK)	smallint unsigned auto_increment	A surrogate key used to uniquely identify each championship.
ChampionshipName	varchar(100)	Championships can be given a name.
ChampionshipYear	year unique	The year of the championship. It is unique because there can't be more than one annual championship each year.

Competition

This table stores archery competition data.

Competition		
Field Name	**Data Type**	**Field Purpose**
CompetitionID (PK)	smallint unsigned auto_increment	A surrogate key used to uniquely identify each competition.
ChampionshipID	smallint unsigned	A competition can be a championship competition. If it's a regular competition, then this field will be null.
CompetitionName	varchar(100)	A competition can be given a name. If not then it will be null.
CompetitionDate	date	Stores the date that the competition took place.

Archer

A table used to store all the archer data.

Archer

Field Name	Data Type	Field Purpose
ArcherID (PK)	int unsigned auto_increment	A surrogate key used to uniquely identify each individual archer.
FirstName	varchar(50)	Stores the first name.
LastName	varchar(50)	Stores the last name.
ArcherAge	tinyint unsigned	Stores the archer’s age.
Gender	enum('M', 'F')	Stores the archer’s gender. Can be either ‘M' or 'F'.

Class

This table stores the different classes in archery. A class is a classification of age and gender.

Class		
Field Name	**Data Type**	**Field Purpose**
ClassName (PK)	varchar(100)	Stores the the class name, i.e., '50+ Male'.
AgeLimitMin	tinyint unsigned	Stores the interval of the set of ages allowed in different class. For example, the interval for under 18 will be x ∈ [0, 18), where x is the age. For open classes, the interval will be x ∈ [0, 255].
AgeLimitMax	tinyint unsigned	
GenderCode	enum('M', 'F')	Stores the archer’s gender. Can be either ‘M' or 'F'.

Category

This table stores all the categories in archery. A category is a combination of a class and a division.

Category		
Field Name	**Data Type**	**Field Purpose**
CategoryName (PK)	varchar(100)	Stores the name of the category, i.e., '50+ Male Longbow'.
ClassName (FK)	varchar(50)	A foreign key that references the table Class. It stores the class of the category.
Division	enum('Recurve', 'Barebow', 'Longbow', 'Compound')	The division stores the bow type of the category.

Round

A table used to store different rounds in archery.

Round		
Field Name	**Data Type**	**Field Purpose**
RoundCode (PK)	varchar(20)	Stores the archery rounds. An example round code is “AA40/1440“.

Range

This table is used to list the number of ends and face sizes of each range for a round. A range is a combination of distance and the number of ends fired at that distance at a particular face target.

When quering this table, the word “range” needs to be surounded with backticks since that’s an sql reserved keyword. For example, `` select * from `range`; ``

\`Range\`		
Field Name	**Data Type**	**Field Purpose**
RangeID (PK)	int unsigned auto_increment	A surrogate key used to uniquely identify each range record.
RoundCode (FK)	varchar(20)	This stores the round that this range record belongs to.
Distance	enum('90', '70', '60', '50', '40', '30', '20', '10')	Stores the distance of a range.
EndCount	enum('5', '6')	Stores the number of ends fired at that distance. The number of ends can either be 5 or 6.
FaceSize	enum('80', '122')	Stores the size of the target at this range. Can be either 80cm or 122cm.

EquivalentRound

A table that stores equivalent rounds. The history of equivalent rounds is recorded in the “Valid“ field.

EquivalentRound		
Field Name	**Data Type**	**Field Purpose**
RoundCode (PK, FK)	varchar(20)	Stores a round code.
EquivalentRoundCode (PK, FK)	varchar(20)	Stores the round code of the round that is equivalent to the first round.
Valid	bit(1)	This field indicates if a pair of equivalent rounds is currently valid.

RoundCategory

This table lists all the categories that are available for each round. If an archer wants to compete in a round, then he must match a category that is available for that round.

RoundCategory		
Field Name	**Data Type**	**Field Purpose**
RoundCode (PK, FK)	varchar(20)	Stores a round code.
CategoryName	varchar(100)	Stores the category that is available for a given 'roundcode'.

RoundResult

This table contains information about a shot round. It doesn’t contain arrow-by-arrow scores; those are found in the 'Score' table.

RoundResult		
Field Name	**Data Type**	**Field Purpose**
RoundResult (PK)	int unsigned auto_increment	A surrogate key used to uniquely identify each round result.
ArcherID	int unsigned	Stores the id of the archer that shot this round.
RoundCode	varchar(20)	Stores the round code of this round.
CategoryName	varchar(100)	Stores the archer’s category for this round.
CompetitionID	smallint unsigned	A round can be a competition round. If not, then this field will be null.
Result	int unsigned	The result is the sum of all ends from all ranges. This must be calculated programmatically.
ResultDate	date	Stores the date in which the round was shot.

Score

The score table will store all the arrow-by-arrow scores from a round stored in 'RoundResult'.

Score		
Field Name	**Data Type**	**Field Purpose**
ScoreID (PK)	int unsigned auto_increment	A surrogate key used to uniquely identify each score.
RoundResultID (FK)	int unsigned	This field stores the round id that

		this score belongs to.
RangeIndex	tinyint unsigned	A round will consist of different ranges. This field stores the index that identifies which range the score belongs to.
EndIndex	tinyint unsigned	Five or six ends are fired at a given distance. This field stores the index that identifies which end the score belongs to.
Arrow1	tinyine unsigned	An end consists of 6 arrow shots. These fields contain the result of each individual arrow. A score can be between 0-10. The individual arrow scores are recorded in descending order, with arrow 1 having the highest score, while arrow 6 has the lowest.
Arrow2		
Arrow3		
Arrow4		
Arrow5		
Arrow6		

Data Creation

Data Creation Overview

Below is a list on how data for each table was created:

- Archer table: 500 records were created in an sql script with a dummy data generator.
- Score table: Approximately 200,000 records are being generated programmatically.
- RoundResult table: Approximately 10,000 records are being generated programmatically.
- Round table: 10 records were created manually in an sql script.
- Category table: 64 records were created manually in an sql script.
- Championship table: 124 records were created in an sql script with a dummy data generator.
- Competition table: 500 records were created in an sql script with a dummy data generator.
- EquivalentRound table: 38 records were created manually in an sql script.
- Class table: 16 records were created manually in an sql script.
- Range table: 39 records were created manually in an sql script.
- RoundCategory table: 136 records were created manually in an sql script.

Sample Insert Queries for each Table

>Archer table

```
insert into archer (FirstName, LastName, ArcherAge, gender)
values ('Celinda', 'Duester', 72, 'F');
```

>Round table

```
insert into round values ('WA90/1440');
```

>Range table

```
insert into `range` values (1, 'WA90/1440', '90', '6', '122');
```

>EquivalentRound table

```
insert into equivalentround values ('WA90/1440', 'AA50/1440', 1);
```

>Class table

```
insert into Class values ('Open Male', 0, 255, 'M');
```

>Category table

```
insert into category values ('Open Male Recurve', 'Open Male', 'Recurve');
```

>RoundCategory table

```
insert into roundcategory values ('Perth', 'Under 16 Female Barebow');
```

>Competition table

```
insert into competition (ChampionshipID, CompetitionName, CompetitionDate)
values (NULL, 'Archery Competition', '1924-12-03');
```

>Championship table

```
insert into championship (ChampionshipName, ChampionshipYear)
values ('Archery Championship', 1901);
```

>RoundResult and Score tables

The records for these tables are being created programmatically with the script described below.

Database Creation PHP Script

This script is used to initialize the archery database, create all the tables, and then populate all the tables with dummy data. To run the script, the required sql files need to be placed in the same directory. In detail, the following operations will be performed:

1. Create the database archery_db.
2. Run the sql script that contains all the 'create table' statements.
3. Run the sql scripts that contain the dummy data. In this step, the sql scripts are executed in the correct order as to avoid any foreign key constraint errors.
4. Generate around 200k-250k scores. In order to generate realistic data, the program ensures that:
   a. The correct number of arrows have been shot for each round. For example, if a round consists of 6 ends fired at 4 different ranges, then 4x6x6 = 144 arrow scores should be recorded.
   b. Each archer competes in the appropriate class, depending on their age and gender.
   c. Archers compete only in rounds where a corresponding category is available.
   d. The date when a round was shot can't be older that the archer who shot it.

```php
<?php

/* Init ------------------------------------------------------------------------ */
$servername = "localhost";
$username = "root";
$password = "";

$conn = new mysqli($servername, $username, $password);

if ($conn->connect_error)
{
    die("Connection failed: " . $conn->connect_error);
}
```

```php

/* Classes ------------------------------------------------------------------------ */
class Archer
{
    public $id;
    public $age;
    public $gender;

    public function __construct($id, $age, $gender)
    {
        $this->id = $id;
        $this->age = $age;
        $this->gender = $gender;
    }

}


class Category
{
    public $name;
    public $age_min;
    public $age_max;
    public $gender;

    public function __construct($name, $age_min, $age_max, $gender)
    {
        $this->name = $name;
        $this->age_min = $age_min;
        $this->age_max = $age_max;
        $this->gender = $gender;
    }

}


class Competition
{
    public $id;
    public $date;

    public function __construct($id, $date)
    {
        $this->id = $id;
        $this->date = $date;
    }

}


/* Functions ---------------------------------------------------------------------- */
function execute_multi_query(&$sql_script): void
{
    global $conn;

    if ($conn->multi_query($sql_script) !== TRUE)
    {
        die("Error executing SQL file: " . $conn->error);
    }

    while(mysqli_more_results($conn))
    {
        mysqli_next_result($conn);
    }
}


function insert_class_record($age_min, $age_max, $class_name, $gender): void
{
    global $conn;

    $query = "insert into class (AgeLimitMin, AgeLimitMax, ClassName, Gender) values ($age_min, $age_max, '$cla
    $result = $conn->query($query);
    if (!$result)
    {
        die("Error: " . $conn->error);
    }
}


function vals_to_array(&$sql_object, $field): array
{
    $arr = array();

    while ($row = $sql_object->fetch_assoc())
    {
        $arr[] = $row[$field];
    }

    return $arr;
}


function get_category(&$archer, &$categories)
{
    $matching_categories = array();

    foreach ($categories as $c)
    {
        if ($archer->age >= $c->age_min && $archer->age <= $c->age_max && $archer->gender === $c->gender)
        {
            $matching_categories[] = $c->name;
        }
    }


    return get_random_array_element($matching_categories);
}

function get_archer_championship_category(&$archer, &$categories, &$championship_categories)
{
    $matching_categories = array();

    foreach ($categories as $c)
    {
        if ($archer->age >= $c->age_min && $archer->age <= $c->age_max && $archer->gender === $c->gender)
```

```php
        {
            if (in_array($c->name, $championship_categories))
            {
                $matching_categories[] = $c->name;
            }
        }
    }

    if (empty($matching_categories))
    {
        return null;
    }

    return get_random_array_element($matching_categories);
}

function get_random_scores(): array
{
    $scores = array();

    for ($i=0; $i < 6; $i++)
    {
        $scores[] = rand(0, 10);
    }

    rsort($scores);

    return $scores;
}


function get_random_date($start_date = '1945-01-01', $end_date = '2024-05-15'): string
{
    $startDate = strtotime($start_date);
    $endDate = strtotime($end_date);
    $randomTimestamp = mt_rand($startDate, $endDate);
    return date('Y-m-d', $randomTimestamp);
}


function get_random_array_element(&$arr)
{
    return $arr[array_rand($arr)];
}


function create_round_result_record($archer_id, $category_code, $round_code, $comp_id, $result_date): int
{
    global $conn;

    $archer_id = (int)$archer_id;

    $insert_query = "insert into roundresult (ArcherID, RoundCode, CategoryName, CompetitionID, Result, ResultD
                  values ($archer_id, '$round_code', '$category_code', $comp_id, NULL, '$result_date')";

    $result = $conn->query($insert_query);
    if (!$result)
    {
        die("create_round_result_record() query failed: " . $conn->error);
    }

    $result_id_query = "select * from roundresult order by roundresultID desc limit 1";
    $result = $conn->query($result_id_query);
    if (!$result)
    {
        die("create_round_result_record() query failed: " . $conn->error);
    }

    return (int)$result->fetch_assoc()["RoundResultID"];
}


function create_round_scores(int $round_result_id, &$round_code, &$rounds): void
{
    global $conn;

    $ranges = $rounds[$round_code];
    $round_result = 0;
    # iterate each range in the round
    for ($range_index = 1; $range_index <= count($ranges); $range_index++)
    {
        $end_count = $ranges[$range_index - 1];

        # iterate each end
        for ($end_index = 1; $end_index <= (int)$end_count; $end_index++)
        {
            $scores = get_random_scores();
            $round_result += array_sum($scores);

            $query = "insert into Score (RoundResultID, RangeIndex, EndIndex, Arrow1, Arrow2, Arrow3, Arrow4, A
                      values ($round_result_id, $range_index, $end_index, $scores[0], $scores[1], $scores[2], $

            $result = $conn->query($query);
            if (!$result)
            {
                die("create_round_scores() query failed: " . $conn->error);
            }
        }
    }

    $update_round_result_query = "update roundresult set Result = $round_result where roundresultid = $round_re
    $result = $conn->query($update_round_result_query);
    if (!$result)
    {
        die("create_round_scores() query failed: " . $conn->error);
    }
}

/* Create archerydb ------------------------------------------------------------------------- */
$create_archery_db_query =
"drop database if exists archerydb; create database archerydb;";

execute_multi_query($create_archery_db_query);

$conn = new mysqli($servername, $username, $password, "archerydb");

```

```php
if ($conn->connect_error)
{
    die("Connection failed: " . $conn->connect_error);
}

set_time_limit(1800);

echo "Connected successfully<br>";


/* SQL Files ------------------------------------------------------------------------- */
$create_tables = file_get_contents("create_tables.sql");
$archer_sql = file_get_contents("archer.sql");
$competition_sql = file_get_contents("competition.sql");
$championship_sql = file_get_contents("championship.sql");
$round_sql = file_get_contents("round.sql");
$range_sql = file_get_contents("range.sql");
$class_sql = file_get_contents("class.sql");
$category_sql = file_get_contents("category.sql");
$equivalent_round_sql = file_get_contents("equivalentround.sql");
$round_category_sql = file_get_contents("roundcategory.sql");


/* Create Tables ------------------------------------------------------------------------- */
execute_multi_query($create_tables);


/* Archer Table ------------------------------------------------------------------------- */
execute_multi_query($archer_sql);

/* Championship Table ------------------------------------------------------------------------- */
execute_multi_query($championship_sql);


/* Competition Table ------------------------------------------------------------------------- */
execute_multi_query($competition_sql);


/* Round Table ------------------------------------------------------------------------- */
execute_multi_query($round_sql);


/* Range Table ------------------------------------------------------------------------- */
execute_multi_query($range_sql);


/* Class Table ------------------------------------------------------------------------- */
execute_multi_query($class_sql);


/* Categorty Table ------------------------------------------------------------------------- */
execute_multi_query($category_sql);


/* Equivalent Round Table ------------------------------------------------------------------------- */
execute_multi_query($equivalent_round_sql);


```

```php
/* Round Category Table ------------------------------------------------------------------------ */
execute_multi_query($round_category_sql);


/* RoundResult and Score Basic Records ------------------------------------------------------------ */
$archer_query = "select ArcherID, ArcherAge, Gender from archer";
$rounds_query = "select * from `range`";

$categories_query =
"select cat.categoryname, cl.agelimitmin, cl.agelimitmax, cl.gender
from category cat
inner join class cl on cat.ClassName = cl.ClassName";

$round_categories_query = "select * from roundcategory";

$archer_records = $conn->query($archer_query);
$round_records = $conn->query($rounds_query);
$category_records = $conn->query($categories_query);
$round_categories_records = $conn->query($round_categories_query);

if (!$archer_records || !$category_records || !$round_records || ! $round_categories_records)
{
    die("Error: " . $conn->error);
}

$archers = array();
$rounds = array();
$categories = array();
$categories_rounds = array();
$round_categories = array();


while ($row = $archer_records->fetch_assoc())
{
    $archers[] = new Archer($row["ArcherID"], $row["ArcherAge"], $row["Gender"]);
}

while ($row = $round_records->fetch_assoc())
{
    $round_code = $row["RoundCode"];
    $end_count = $row["EndCount"];

    if (array_key_exists($round_code, $rounds))
    {
        $rounds[$round_code][] = $end_count;
    }
    else
    {
        $rounds[$round_code] = array($end_count);
    }
}

while ($row = $category_records->fetch_assoc())
{
    $categories[] = new Category($row["categoryname"], $row["agelimitmin"], $row["agelimitmax"], $row["gender"]
}

while ($row = $round_categories_records->fetch_assoc())
```

```
{
    $category_name = $row["CategoryName"];
    $round_code = $row["RoundCode"];

    if (array_key_exists($category_name, $categories_rounds))
    {
        $categories_rounds[$category_name][] = $round_code;
    }
    else
    {
        $categories_rounds[$category_name] = array($round_code);
    }

    if (array_key_exists($round_code, $round_categories))
    {
        $round_categories[$round_code][] = $category_name;
    }
    else
    {
        $round_categories[$round_code] = array($category_name);
    }
}

$round_keys = array_keys($rounds);

# create a championship for testing
$championship_round = 'Melbourne';
$championship_categories = $round_categories[$championship_round];
$competition_id = 4;
$competition_date = '2023-01-27';

foreach ($archers as $archer)
{
    $archer_birth_date = date("Y-m-d", strtotime("-$archer->age years"));

    # create 20 records for each archer
    for ($i = 0; $i < 20; $i++)
    {
        $archer_category = get_category($archer, $categories);
        $result_date = get_random_date($archer_birth_date);
        $round_code = get_random_array_element($categories_rounds[$archer_category]);

        $round_result_id = create_round_result_record($archer->id, $archer_category, $round_code, "NULL", $resu

        create_round_scores($round_result_id, $round_code, $rounds);
    }

    $champ_category = get_archer_championship_category($archer, $categories, $championship_categories);

    if ($champ_category)
    {
        $round_result_id = create_round_result_record($archer->id, $champ_category, $championship_round, $compe
        create_round_scores($round_result_id, $championship_round, $rounds);
    }
}


/* Terminate ------------------------------------------------------------------------ */
```

```php
$conn->close();

echo "EXIT_SUCCESS<br>";

```

Use Cases

Introduction

From studying the client’s business scenario, a set of requirements have been identified:

1. Archers need to be able to look at their score.
2. Number of scores should be able to be restricted by date range and by type of round.
3. Archers need to be able to look up definitions of rounds and equivalent rounds.
4. Archers need to be able to look up various metrics of a competition, e.g., the totals of all arrows of the round shot.
5. Archers need to be able to look up various metrics of a championship, e.g., the winner of the championship in each category.
6. Archers need to be able to look up their best score for a particular round.
7. The club's best score for a round and the archer who shot it should be an available lookup.
8. The scores have to contain arrow-by-arrow scores. Each arrow score has to be able to be identified in terms of which end it belongs to. Each end has to be identified as to its position in the round score. Within an end, arrows are always recorded highest to lowest arrow score.
9. The recorder has to be able to enter new archers, new rounds and new competitions.
10. Some of the scores have to be able to be linked to a competition. Some competitions have to be able to be identified as part of a club championship.
11. The database has to have all the information needed to identify the archer's division.
12. Category can be identified when the bow type is absent on user input.
13. The equivalent rounds have to be time-dependent, and become invalidated when they change by Archery Australia.

We have developed a list of SQL queries that aim to satisfy each and every one of the requirements listed above.

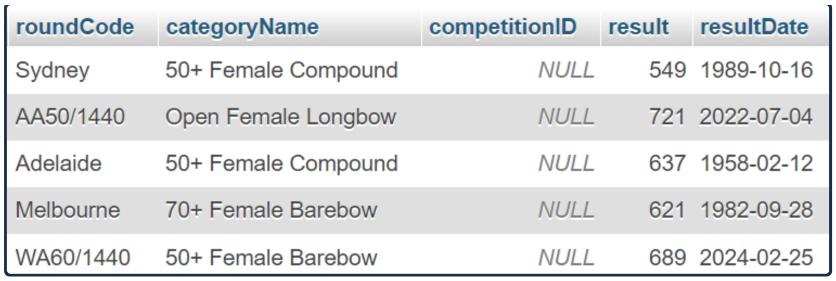
SQL Queries

1. Archers need to be able to look at their score.

This query selects all the round scores of archer with archerID = 1. The column ‘result' is the sum of all arrows shot at different ranges in a round. In the current db design, 'result’ is meant to be calulated programmatically.

```sql
SELECT roundCode, result, resultDate
FROM RoundResult
WHERE archerID = 1;
```

Output



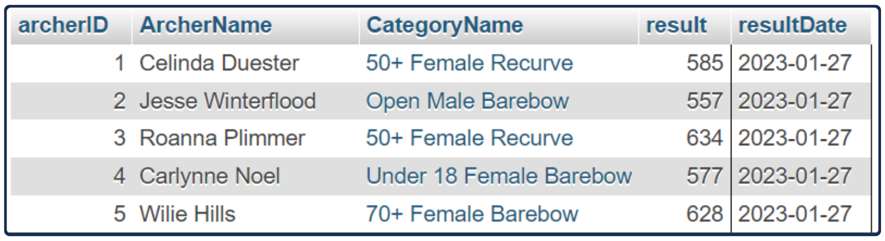
roundCode	categoryName	competitionID	result	resultDate
Sydney	50+ Female Compound	NULL	549	1989-10-16
AA50/1440	Open Female Longbow	NULL	721	2022-07-04
Adelaide	50+ Female Compound	NULL	637	1958-02-12
Melbourne	70+ Female Barebow	NULL	621	1982-09-28
WA60/1440	50+ Female Barebow	NULL	689	2024-02-25

2. Number of scores should be able to be restricted by date range and by type of round.

This query selects all the round scores of the ‘Melbourne' round between 2022 and 2024.

```sql
SELECT rr.archerID, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName,
    rr.result, rr.resultDate
FROM RoundResult rr
INNER JOIN archer a on rr.ArcherID = a.ArcherID
WHERE roundCode = 'Melbourne' AND resultDate BETWEEN '2022-01-01' AND '2024-01-01';
```

Output



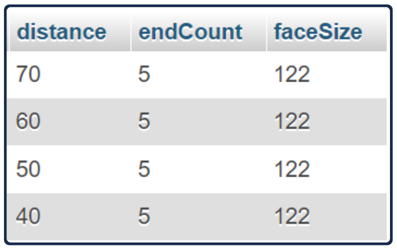
archerID	ArcherName	CategoryName	result	resultDate
1	Celinda Duester	50+ Female Recurve	585	2023-01-27
2	Jesse Winterflood	Open Male Barebow	557	2023-01-27
3	Roanna Plimmer	50+ Female Recurve	634	2023-01-27
4	Carlynne Noel	Under 18 Female Barebow	577	2023-01-27
5	Wilie Hills	70+ Female Barebow	628	2023-01-27

3. Archers need to be able to look up definitions of rounds and equivalent rounds.

a. This query lists all the information about the 'Melbourne' round.

```sql
SELECT distance, endCount, faceSize
FROM `range`
WHERE roundCode = 'Melbourne';
```

Output

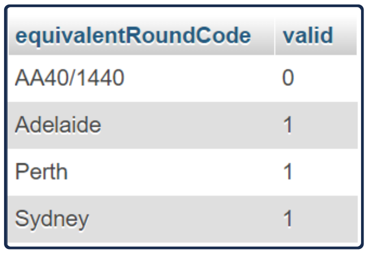


distance	endCount	faceSize
70	5	122
60	5	122
50	5	122
40	5	122

b. This query returns all the rounds that are equivalent to the 'Melbourne' round.

```sql
SELECT equivalentRoundCode, valid
FROM equivalentround
WHERE roundCode = 'Melbourne';
```

Output



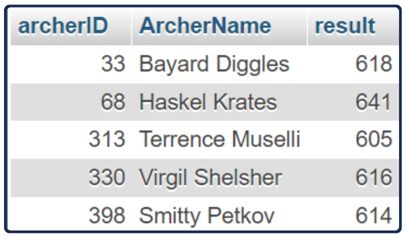
equivalentRoundCode	valid
AA40/1440	0
Adelaide	1
Perth	1
Sydney	1

4. Archers need to be able to look up various metrics of a competition.

This query list the results of the '50+ Male Recurve' category from the competion with ID = 4.

```sql
SELECT rr.archerID, CONCAT(a.FirstName, ' ', a.LastName) AS ArcherName, rr.result
FROM roundResult rr
INNER JOIN archer a ON rr.ArcherID = a.ArcherID
WHERE competitionID = 4 and rr.CategoryName = '50+ Male Recurve';
```

Output



archerID	ArcherName	result
33	Bayard Diggles	618
68	Haskel Krates	641
313	Terrence Muselli	605
330	Virgil Shelsher	616
398	Smitty Petkov	614

5. Archers need to be able to look up various metrics of a championship.

This query lists the highest scores for each category in the 2023 championship.

```sql
SELECT rr.CategoryName, CONCAT(a.FirstName, ' ', a.LastName) AS ArcherName, rr.Result
FROM roundresult rr
INNER JOIN archer a ON rr.ArcherID = a.ArcherID
INNER JOIN competition comp ON rr.CompetitionID = comp.CompetitionID
INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID
INNER JOIN (
    SELECT rr.categoryName as category, MAX(rr.result) AS score
    FROM roundresult rr
    INNER JOIN competition comp ON rr.CompetitionID = comp.CompetitionID
    INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID
    WHERE champ.ChampionshipYear = 2023
    GROUP BY rr.CategoryName) AS sub
        ON rr.CategoryName = sub.category AND rr.Result = sub.score
WHERE champ.ChampionshipYear = 2023
```

Output

CategoryName	ArcherName	Result
70+ Male Compound	Harald Guilleton	623
60+ Male Barebow	Stacee Agate	637
50+ Male Recurve	Haskel Krates	641
50+ Female Recurve	Rosmunda Ablitt	694
Under 18 Female Barebow	Bunni Garvill	640
70+ Female Barebow	Audrie Meneely	688

6. Archers need to be able to look up their best score for a particular round.

This query lists the highest round results of archer with ID = 1;

```
SELECT roundCode, MAX(result) as Result, resultDate
FROM roundresult
WHERE ArcherID = 1
GROUP BY roundCode;
```

Output

roundCode	Result	resultDate
AA50/1440	741	2022-07-04
Adelaide	637	1958-02-12
Brisbane	549	2023-05-14
Melbourne	622	1982-09-28

7. The club's best score for a round and the archer who shot it should be an available lookup.

This query lists the highest round score shot for each round and the archer who shot it.

```
SELECT rr.RoundCode, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName,
    rr.Result, rr.ResultDate
FROM roundresult rr
INNER JOIN archer a ON rr.ArcherID = a.ArcherID
WHERE (rr.RoundCode, rr.Result) IN (
    SELECT RoundCode, MAX(Result) AS HighestRoundScoreEver
    FROM roundresult
    GROUP BY RoundCode)
```

Output

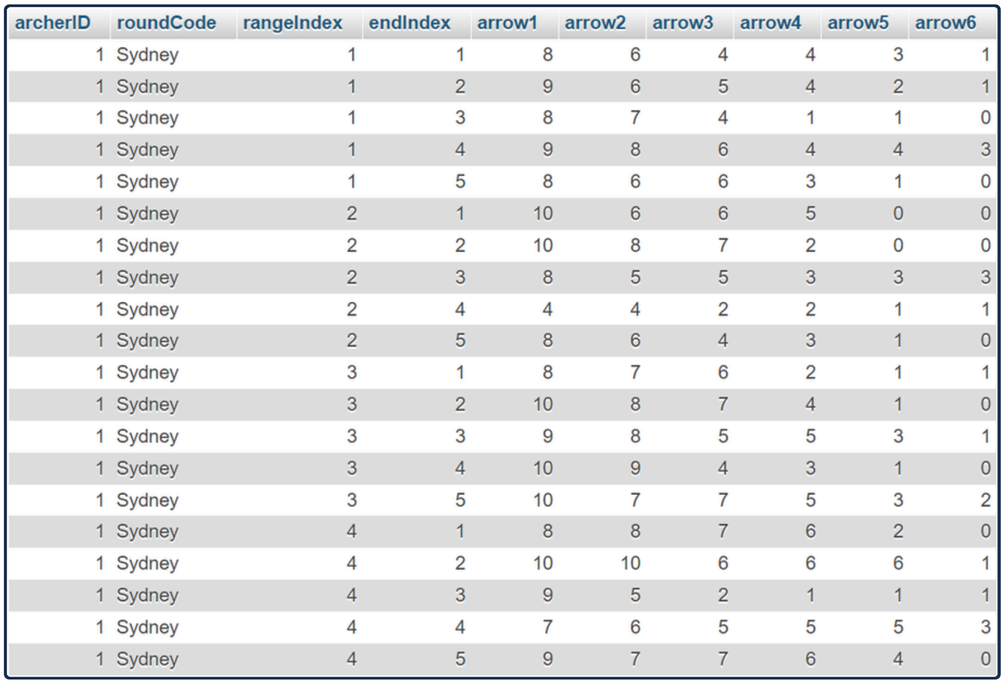
RoundCode	ArcherName	Result	ResultDate
WA70/1440	Nedda Benitez	833	1967-07-28
Perth	Jacky Teaz	513	2012-08-21
AA40/1440	Kip Pennino	795	2019-01-06
Brisbane	Waneta Poznanski	743	1975-04-19
Melbourne	Lorna Faas	720	2004-09-06

8. The scores have to contain arrow-by-arrow scores. Each arrow score has to be able to be identified in terms of which end it belongs to. Each end has to be identified as to its position in the round score. Within an end, arrows are always recorded highest to lowest arrow score.

This queries lists arrow by arrow scores of the round with ID = 1, along with the archer that shot it.

```sql
SELECT rr.archerID, rr.roundCode, s.rangeIndex, s.endIndex,
     s.arrow1, s.arrow2, s.arrow3, s.arrow4, s.arrow5, s.arrow6
FROM roundresult rr
INNER JOIN score s ON rr.RoundResultID = s.RoundResultID
WHERE rr.RoundResultID = 1;
```

Output



archerID	roundCode	rangeIndex	endIndex	arrow1	arrow2	arrow3	arrow4	arrow5	arrow6
1	Sydney	1	1	8	6	4	4	3	1
1	Sydney	1	2	9	6	5	4	2	1
1	Sydney	1	3	8	7	4	1	1	0
1	Sydney	1	4	9	8	6	4	4	3
1	Sydney	1	5	8	6	6	3	1	0
1	Sydney	2	1	10	6	6	5	0	0
1	Sydney	2	2	10	8	7	2	0	0
1	Sydney	2	3	8	5	5	3	3	3
1	Sydney	2	4	4	4	2	2	1	1
1	Sydney	2	5	8	6	4	3	1	0
1	Sydney	3	1	8	7	6	2	1	1
1	Sydney	3	2	10	8	7	4	1	0
1	Sydney	3	3	9	8	5	5	3	1
1	Sydney	3	4	10	9	4	3	1	0
1	Sydney	3	5	10	7	7	5	3	2
1	Sydney	4	1	8	8	7	6	2	0
1	Sydney	4	2	10	10	6	6	6	1
1	Sydney	4	3	9	5	2	1	1	1
1	Sydney	4	4	7	6	5	5	5	3
1	Sydney	4	5	9	7	7	6	4	0

9. The recorder has to be able to enter new archers, new rounds and new competitions.

a. New archer record

```sql
INSERT INTO archer (FirstName, LastName, ArcherAge, Gender)
VALUES ('John', 'Iliadis', 22, 'M');
```

b. New round record

```sql
INSERT INTO round VALUES ('Darwin');

INSERT INTO `range` (RoundCode, Distance, EndCount, FaceSize)
VALUES ('Darwin', '10', '6', '80');
INSERT INTO `range` (RoundCode, Distance, EndCount, FaceSize)
VALUES ('Darwin', '20', '6', '80');
INSERT INTO `range` (RoundCode, Distance, EndCount, FaceSize)
VALUES ('Darwin', '30', '6', '80');
INSERT INTO `range` (RoundCode, Distance, EndCount, FaceSize)
VALUES ('Darwin', '40', '6', '80');
```

c. New competition record

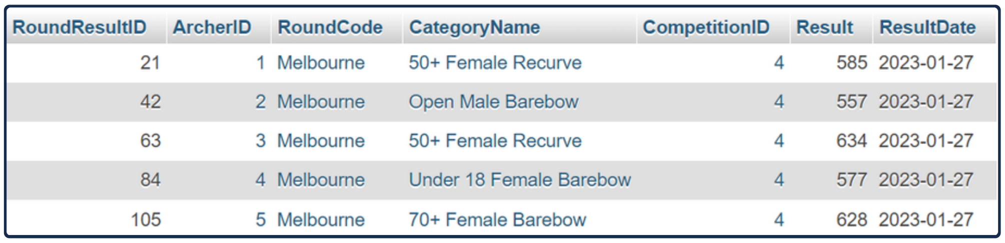
```sql
INSERT INTO competition (championshipID, CompetitionName, CompetitionDate)
VALUES (NULL, 'New competition', '2024-05-15');
```

10. Some of the scores have to be able to be linked to a competition. Some competitions have to be able to be identified as part of a club championship.

a. This query lists all the competition round results

```sql
SELECT *
FROM roundresult
WHERE competitionID IS NOT NULL;
```

Output

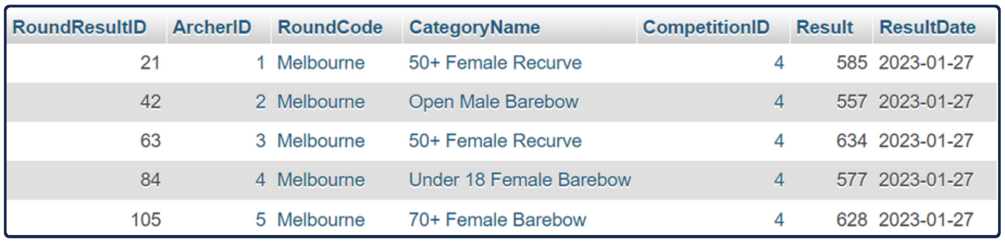


RoundResultID	ArcherID	RoundCode	CategoryName	CompetitionID	Result	ResultDate
21	1	Melbourne	50+ Female Recurve	4	585	2023-01-27
42	2	Melbourne	Open Male Barebow	4	557	2023-01-27
63	3	Melbourne	50+ Female Recurve	4	634	2023-01-27
84	4	Melbourne	Under 18 Female Barebow	4	577	2023-01-27
105	5	Melbourne	70+ Female Barebow	4	628	2023-01-27

b. This query lists all the championship round results.

```sql
SELECT rr.*
FROM roundresult rr
INNER JOIN competition c on rr.CompetitionID = c.CompetitionID
WHERE c.ChampionshipID IS NOT NULL;
```

Output



RoundResultID	ArcherID	RoundCode	CategoryName	CompetitionID	Result	ResultDate
21	1	Melbourne	50+ Female Recurve	4	585	2023-01-27
42	2	Melbourne	Open Male Barebow	4	557	2023-01-27
63	3	Melbourne	50+ Female Recurve	4	634	2023-01-27
84	4	Melbourne	Under 18 Female Barebow	4	577	2023-01-27
105	5	Melbourne	70+ Female Barebow	4	628	2023-01-27

11. The database has to have all the information needed to identify the archer's division.

This query lists all the available categories that the archer with ID = 1 can compete in for the round 'WA70/1440'. The division is listed in the category name. The class as well.

```sql
SELECT sq.categoryName
FROM (SELECT rc.roundcode, rc.categoryName, cl.agelimitmin, cl.agelimitmax, cl.gender
      FROM roundcategory rc
      INNER JOIN category cat ON rc.CategoryName = cat.CategoryName
      INNER JOIN class cl ON cat.ClassName = cl.ClassName
      WHERE rc.RoundCode = 'WA70/1440') AS sq
INNER JOIN archer a ON a.ArcherAge BETWEEN sq.agelimitmin AND sq.agelimitmax
```

```sql
WHERE a.ArcherID = 1 AND sq.gender = a.gender;
```

Output

categoryName
Open Female Compound
Open Female Recurve

12. Category can be identified when the bow type is absent on user input.

This query lists all the categories that the archer with ID = 1 can compete in.

```sql
SELECT sq.categoryName
FROM (SELECT cat.CategoryName, cl.AgeLimitMin, cl.AgeLimitMax, cl.Gender
      FROM category cat
      INNER JOIN class cl ON cat.ClassName = cl.ClassName) AS sq
INNER JOIN archer a ON a.ArcherAge BETWEEN sq.agelimitmin AND sq.agelimitmax
WHERE a.ArcherID = 1 AND sq.gender = a.gender;
```

Output

CategoryName
50+ Female Barebow
50+ Female Compound
50+ Female Longbow
50+ Female Recurve

13. The equivalent rounds have to be time-dependent, and become invalidated when they change by Archery Australia.

This query returns all the currently valid equivalent rounds to the 'Melbourne' round.

```sql
SELECT equivalentRoundCode
FROM equivalentround
WHERE roundCode = 'Melbourne' and valid = 1;
```

Output

equivalentRoundCode
Adelaide
Perth
Sydney

Transactions

In SQL, a transaction is a set of queries that get executed as a single unit of work. Within a transaction, all the queries must run succesfully for the changes to be applied. This ensures that the database is always in a consistent state.

Below is an example transaction where the “Sydney“ round is defined, as well as all its equivalent rounds.

```sql
START TRANSACTION;
    INSERT INTO round VALUES ('Sydney');
    INSERT INTO `range` VALUES (31,'Sydney', '90', '5', '122');
    INSERT INTO `range` VALUES (32,'Sydney', '70', '5', '122');
    INSERT INTO `range` VALUES (33,'Sydney', '60', '5', '122');
    INSERT INTO `range` VALUES (34,'Sydney', '50', '5', '122');
    INSERT INTO equivalentround VALUES ('Sydney', 'Melbourne', 1);
    INSERT INTO equivalentround VALUES ('Sydney', 'WA60/1440', 1);
    INSERT INTO equivalentround VALUES ('Sydney', 'AA40/1440', 0);
COMMIT;
```

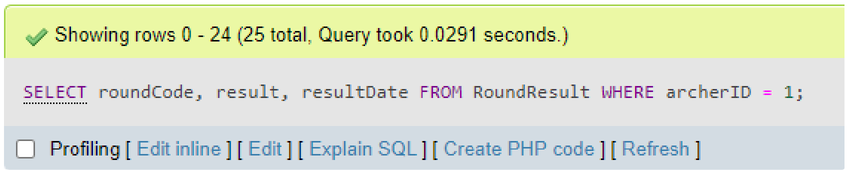
In this project, we are mainly focused on the database layer of the system. Transaction operations are usually implemented in the application layer, so they are not included as part of the current solution.

Indexing

Score lookup

```sql
SELECT roundCode, result, resultDate
FROM RoundResult
WHERE archerID = 1;
```

Before



✓ Showing rows 0 - 24 (25 total, Query took 0.0291 seconds.)

```sql
SELECT roundCode, result, resultDate FROM RoundResult WHERE archerID = 1;
```

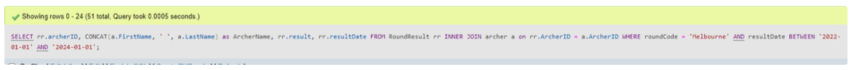
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

No index is needed as archerID is a foreign key for roundResult meaning it is already indexed by default

Selecting scores based on the date

```sql
SELECT rr.archerID, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName,
    rr.result, rr.resultDate
FROM RoundResult rr
INNER JOIN archer a on rr.ArcherID = a.ArcherID
WHERE roundCode = 'Melbourne' AND resultDate BETWEEN '2022-01-01' AND '2024-01-01';
```

Before

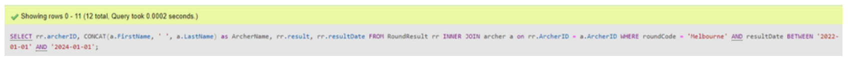


✓ Showing rows 0 - 24 (51 total, Query took 0.0005 seconds.)

```sql
SELECT rr.archerID, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName, rr.result, rr.resultDate FROM RoundResult rr INNER JOIN archer a on rr.ArcherID = a.ArcherID WHERE roundCode = 'Melbourne' AND resultDate BETWEEN '2022-01-01' AND '2024-01-01';
```

```sql
CREATE INDEX idx_resultDate ON RoundResult (resultDate);
```

After



✓ Showing rows 0 - 11 (12 total, Query took 0.0002 seconds.)

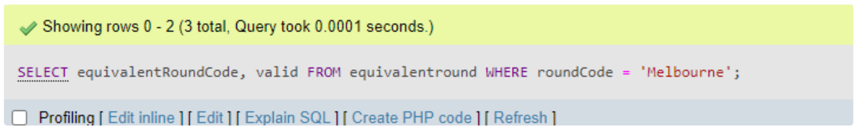
```sql
SELECT rr.archerID, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName, rr.result, rr.resultDate FROM RoundResult rr INNER JOIN archer a on rr.ArcherID = a.ArcherID WHERE roundCode = 'Melbourne' AND resultDate BETWEEN '2022-01-01' AND '2024-01-01';
```

The index enables better row retrieval as there is archers per event as well as multiple different event dates in the event date range that has been set.

Look Up round and equivalent round

```sql
SELECT equivalentRoundCode, valid
FROM equivalentround
WHERE roundCode = 'Melbourne';
```

Before



✓ Showing rows 0 - 2 (3 total, Query took 0.0001 seconds.)

```sql
SELECT equivalentRoundCode, valid FROM equivalentround WHERE roundCode = 'Melbourne';
```

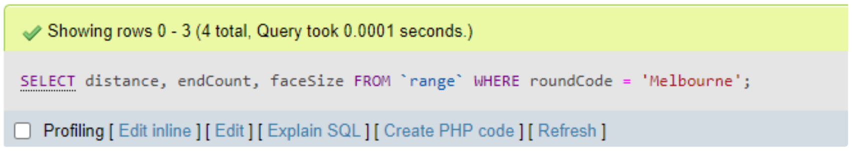
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

No Index is needed for this query as round code is a foreign key for equivalentRound and is set to an index by default.

Look Up round from round code

```sql
SELECT distance, endCount, faceSize
```

```
FROM range
WHERE roundCode = 'Melbourne';
```



Showing rows 0 - 3 (4 total, Query took 0.0001 seconds.)

```
SELECT distance, endCount, faceSize FROM `range` WHERE roundCode = 'Melbourne';
```

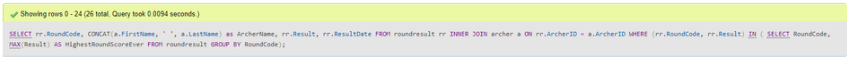
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

This query like the previous one are already indexed as roundCode is a foreign key.

Look Up Best score for each round and who shot it

```
SELECT rr.RoundCode, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName,
    rr.Result, rr.ResultDate
FROM roundresult rr
INNER JOIN archer a ON rr.ArcherID = a.ArcherID
WHERE (rr.RoundCode, rr.Result) IN (
    SELECT RoundCode, MAX(Result) AS HighestRoundScoreEver
    FROM roundresult
    GROUP BY RoundCode)
```

Before

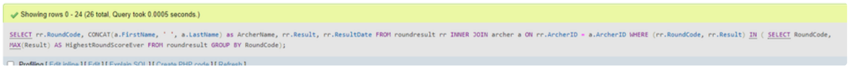


Showing rows 0 - 24 (26 total, Query took 0.0094 seconds.)

```
SELECT rr.RoundCode, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName, rr.Result, rr.ResultDate FROM roundresult rr INNER JOIN archer a ON rr.ArcherID = a.ArcherID WHERE (rr.RoundCode, rr.Result) IN ( SELECT RoundCode, MAX(Result) AS HighestRoundScoreEver FROM roundresult GROUP BY RoundCode);
```

```
CREATE INDEX idx_RoundCode_Result ON roundresult (Result);
```

After



Showing rows 0 - 24 (26 total, Query took 0.0005 seconds.)

```
SELECT rr.RoundCode, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName, rr.Result, rr.ResultDate FROM roundresult rr INNER JOIN archer a ON rr.ArcherID = a.ArcherID WHERE (rr.RoundCode, rr.Result) IN ( SELECT RoundCode, MAX(Result) AS HighestRoundScoreEver FROM roundresult GROUP BY RoundCode);
```

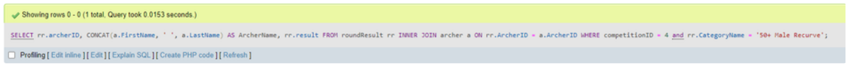
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

The index helps as it makes it easier for the query to search for the results in the other tables.

Look up various metrics of competitions

```
SELECT rr.archerID, CONCAT(a.FirstName, ' ', a.LastName) AS ArcherName, rr.result
FROM roundResult rr
INNER JOIN archer a ON rr.ArcherID = a.ArcherID
WHERE competitionID = 4 and rr.CategoryName = '50+ Male Recurve';
```

Before



Showing rows 0 - 0 (1 total, Query took 0.0153 seconds.)

```
SELECT rr.archerID, CONCAT(a.FirstName, ' ', a.LastName) AS ArcherName, rr.result FROM roundResult rr INNER JOIN archer a ON rr.ArcherID = a.ArcherID WHERE competitionID = 4 and rr.CategoryName = '50+ Male Recurve';
```

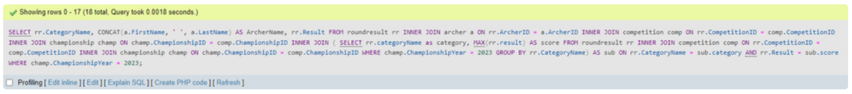
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

The category name is already indexed as well as the competitionID as they are both foreign keys.

Look Up various metrics of championship

```
SELECT rr.CategoryName, CONCAT(a.FirstName, ' ', a.LastName) AS ArcherName, rr.Result
FROM roundresult rr
INNER JOIN archer a ON rr.ArcherID = a.ArcherID
INNER JOIN competition comp ON rr.CompetitionID = comp.CompetitionID
INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID
INNER JOIN (
    SELECT rr.categoryName as category, MAX(rr.result) AS score
    FROM roundresult rr
    INNER JOIN competition comp ON rr.CompetitionID = comp.CompetitionID
    INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID
    WHERE champ.ChampionshipYear = 2023
    GROUP BY rr.CategoryName) AS sub
        ON rr.CategoryName = sub.category AND rr.Result = sub.score
WHERE champ.ChampionshipYear = 2023
```

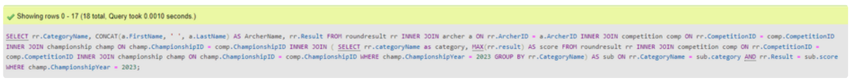
Before



```
Showing rows 0 - 17 (18 total, Query took 0.0018 seconds.)

SELECT rr.CategoryName, CONCAT(a.FirstName, ' ', a.LastName) AS ArcherName, rr.Result FROM roundresult rr INNER JOIN archer a ON rr.ArcherID = a.ArcherID INNER JOIN competition comp ON rr.CompetitionID = comp.CompetitionID
INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID INNER JOIN ( SELECT rr.categoryName as category, MAX(rr.result) AS score FROM roundresult rr INNER JOIN competition comp ON rr.CompetitionID =
comp.CompetitionID INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID WHERE champ.ChampionshipYear = 2023 GROUP BY rr.CategoryName) AS sub ON rr.CategoryName = sub.category AND rr.Result = sub.score
WHERE champ.ChampionshipYear = 2023;

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
```

```
CREATE INDEX idx_champ_ChampionshipYear ON championship (ChampionshipYear);
```

After



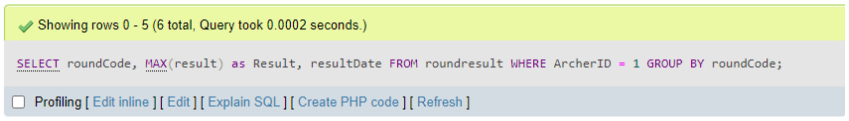
```
Showing rows 0 - 17 (18 total, Query took 0.0010 seconds.)

SELECT rr.CategoryName, CONCAT(a.FirstName, ' ', a.LastName) AS ArcherName, rr.Result FROM roundresult rr INNER JOIN archer a ON rr.ArcherID = a.ArcherID INNER JOIN competition comp ON rr.CompetitionID = comp.CompetitionID
INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID INNER JOIN ( SELECT rr.categoryName as category, MAX(rr.result) AS score FROM roundresult rr INNER JOIN competition comp ON rr.CompetitionID =
comp.CompetitionID INNER JOIN championship champ ON champ.ChampionshipID = comp.ChampionshipID WHERE champ.ChampionshipYear = 2023 GROUP BY rr.CategoryName) AS sub ON rr.CategoryName = sub.category AND rr.Result = sub.score
WHERE champ.ChampionshipYear = 2023;
```

The index helps in the joining of the tables by making it easier for them to filter each row.

Look Up Best Score For Round

```
SELECT roundCode, MAX(result) as Result, resultDate
FROM roundresult
WHERE ArcherID = 1
GROUP BY roundCode;
```

Before



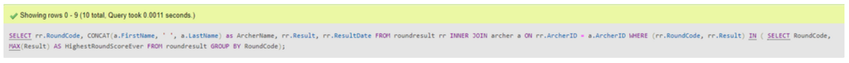
```
Showing rows 0 - 5 (6 total, Query took 0.0002 seconds.)

SELECT roundCode, MAX(result) as Result, resultDate FROM roundresult WHERE ArcherID = 1 GROUP BY roundCode;

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
```

No new index is required as an index to help search by archerID is already made by default as well as round code.

Look Up Club Best Score For Round

```
 SELECT rr.RoundCode, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName,
    rr.Result, rr.ResultDate
FROM roundresult rr
INNER JOIN archer a ON rr.ArcherID = a.ArcherID
WHERE (rr.RoundCode, rr.Result) IN (
    SELECT RoundCode, MAX(Result) AS HighestRoundScoreEver
    FROM roundresult
    GROUP BY RoundCode)
```

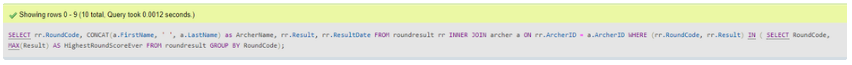
Before



```
Showing rows 0 - 9 (10 total, Query took 0.0011 seconds.)

SELECT rr.RoundCode, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName, rr.Result, rr.ResultDate FROM roundresult rr INNER JOIN archer a ON rr.ArcherID = a.ArcherID WHERE (rr.RoundCode, rr.Result) IN ( SELECT RoundCode,
MAX(Result) AS HighestRoundScoreEver FROM roundresult GROUP BY RoundCode);
```

```
CREATE INDEX idx_rr_archer_round_result ON roundresult (Result, ResultDate);
```

After



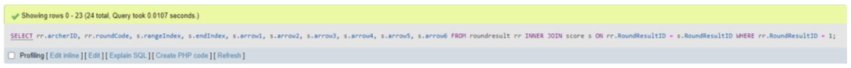
```
Showing rows 0 - 9 (10 total, Query took 0.0012 seconds.)

SELECT rr.RoundCode, CONCAT(a.FirstName, ' ', a.LastName) as ArcherName, rr.Result, rr.ResultDate FROM roundresult rr INNER JOIN archer a ON rr.ArcherID = a.ArcherID WHERE (rr.RoundCode, rr.Result) IN ( SELECT RoundCode,
MAX(Result) AS HighestRoundScoreEver FROM roundresult GROUP BY RoundCode);
```

The speed of the query does not improve, this may be to do with the query and how it is looking for very specific outputs.

Arrow By Arrow Score

```sql
SELECT rr.archerID, rr.roundCode, s.rangeIndex, s.endIndex,
     s.arrow1, s.arrow2, s.arrow3, s.arrow4, s.arrow5, s.arrow6
FROM roundresult rr
INNER JOIN score s ON rr.RoundResultID = s.RoundResultID
WHERE rr.RoundResultID = 1;
```

before



```
Showing rows 0 - 23 (24 total, Query took 0.0107 seconds.)
SELECT rr.archerID, rr.roundCode, s.rangeIndex, s.endIndex, s.arrow1, s.arrow2, s.arrow3, s.arrow4, s.arrow5, s.arrow6 FROM roundresult rr INNER JOIN score s ON rr.RoundResultID = s.RoundResultID WHERE rr.RoundResultID = 1;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

The roundResutlID is already indexed as it is a foreign key.

In conclusion some of the indexes helped the database with the indexes below being the helpful ones used. The Indexes below are the indexes that will be used in the database as they are all useful at grouping rows to make it easier for queries to look up all rows relevant to the conditions.

Indexes used:

```sql
CREATE INDEX idx_RoundCode_Result ON roundresult (Result);
CREATE INDEX idx_resultDate ON RoundResult (resultDate);
CREATE INDEX idx_champ_ChampionshipYear ON championship (ChampionshipYear);
```