MARIO KART
Database

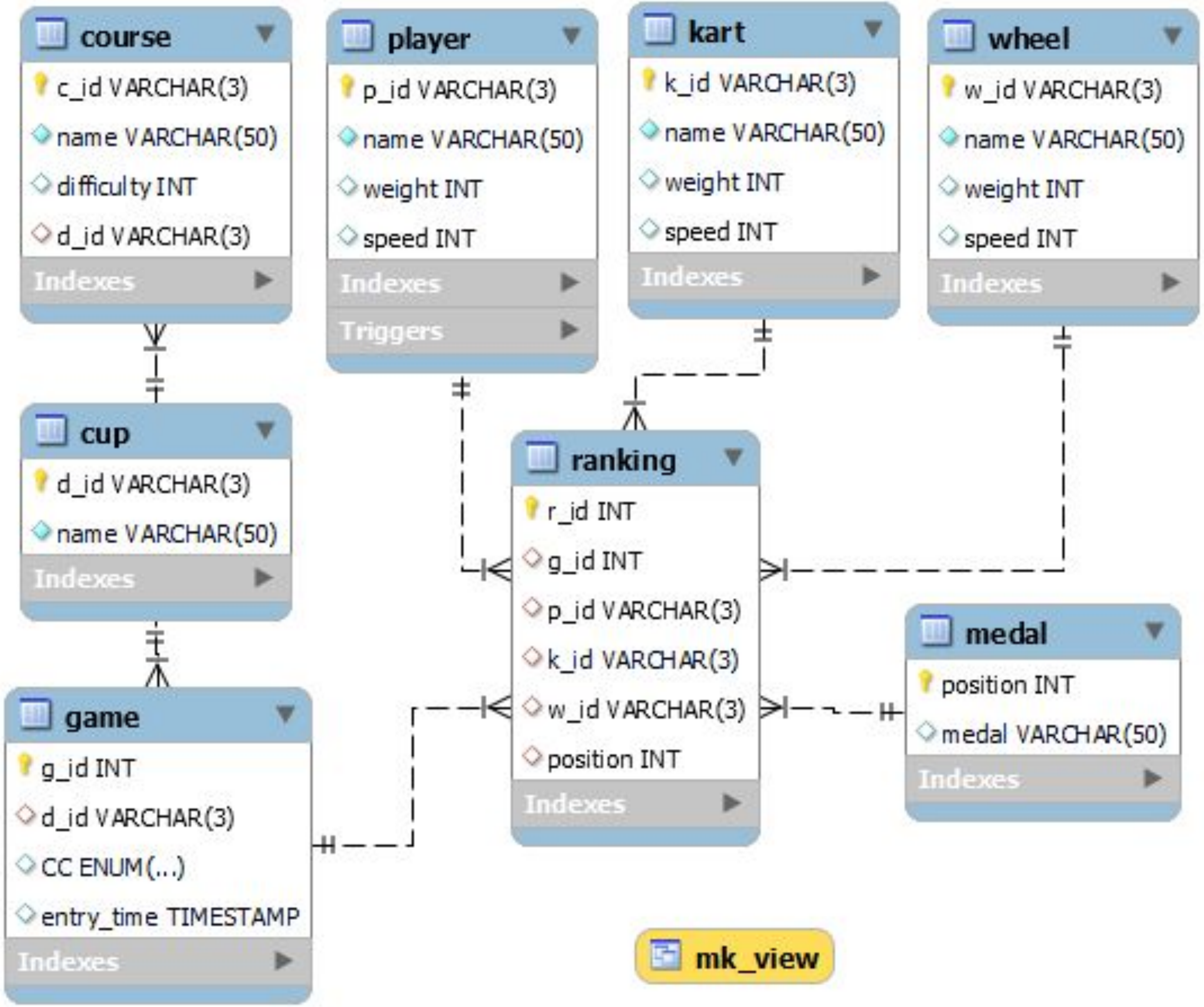
Isabel Doherty and Helen Youssef

CODE FIRST GIRLS

ER diagram

Our database via ER diagram

- Relations between PKs and FKs shown by lines
- This database obeys 3rd Normal Form as each non-key attribute depends only on the key



course
- c_id VARCHAR(3)
- name VARCHAR(50)
- difficulty INT
- d_id VARCHAR(3)
- Indexes

player
- p_id VARCHAR(3)
- name VARCHAR(50)
- weight INT
- speed INT
- Indexes
- Triggers

kart
- k_id VARCHAR(3)
- name VARCHAR(50)
- weight INT
- speed INT
- Indexes

wheel
- w_id VARCHAR(3)
- name VARCHAR(50)
- weight INT
- speed INT
- Indexes

cup
- d_id VARCHAR(3)
- name VARCHAR(50)
- Indexes

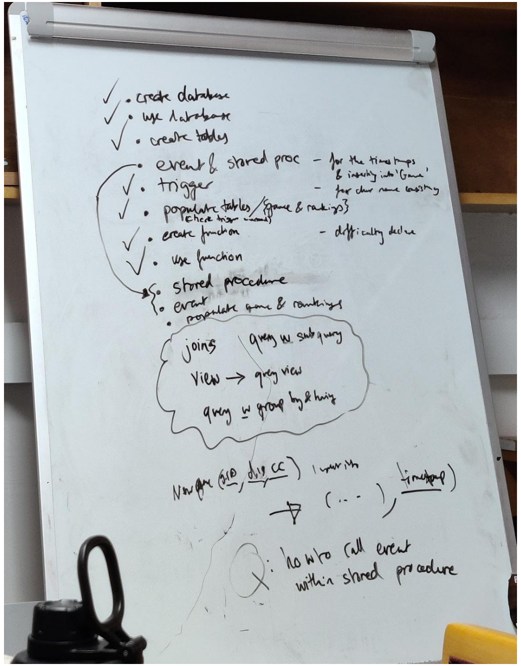
ranking
- r_id INT
- g_id INT
- p_id VARCHAR(3)
- k_id VARCHAR(3)
- w_id VARCHAR(3)
- position INT
- Indexes

medal
- position INT
- medal VARCHAR(50)
- Indexes

game
- g_id INT
- d_id VARCHAR(3)
- CC ENUM(...)
- entry_time TIMESTAMP
- Indexes

mk_view

Contents



✓ • create database
✓ • use database
✓ • create tables
• event & stored proc – for the timestamps & inserting into 'Game'
✓ • trigger – for char name consisting
✓ • populate tables / {game & rankings}
(check trigger [illegible])
✓ • create function – difficulty declare
✓ • use function
• stored procedure
• event
• populate game & rankings

joins / query w subquery
view → query view
query w group by & having

New game (gID, diff, cc) [illegible]
→ (...), timestamp)

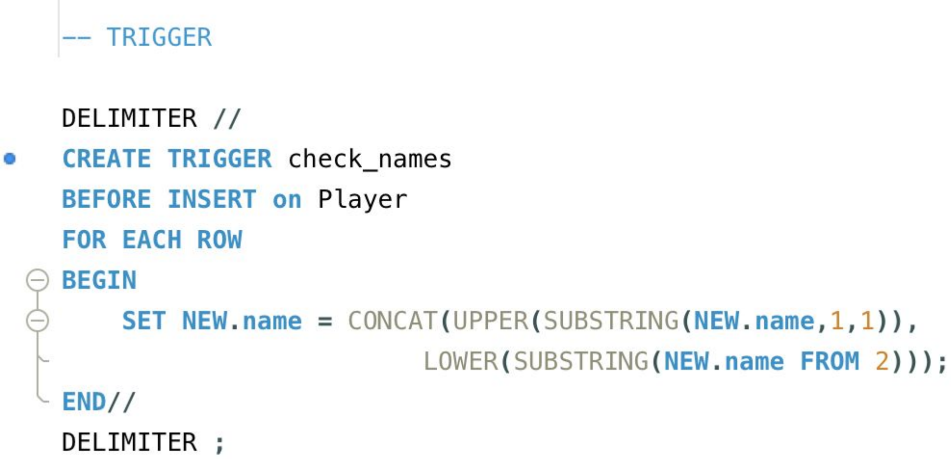
Q: how to call event within stored procedure

- Create Tables
- Create a Trigger
- Populate Tables (w/o game & ranking)
- Create a Function
- Stored Procedure (used to populate game & ranking)
- Event
- Joins with ‘group by’ and ‘having’
- Query with subquery
- View and query

Trigger

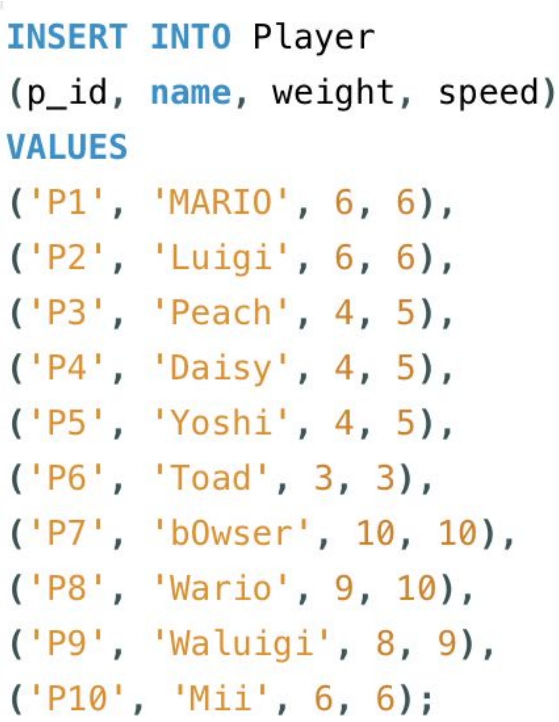
Our trigger was used to ensure the output of the character names would be consistent.

Trigger Code:



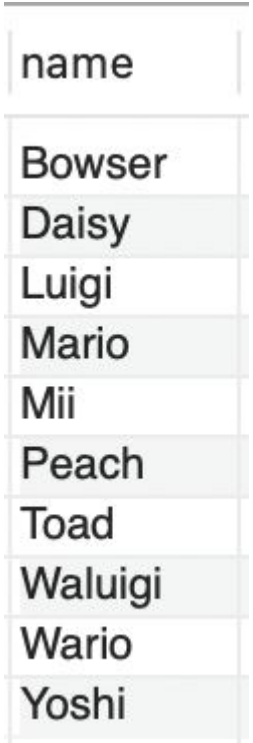
```
-- TRIGGER

DELIMITER //
CREATE TRIGGER check_names
BEFORE INSERT on Player
FOR EACH ROW
BEGIN
    SET NEW.name = CONCAT(UPPER(SUBSTRING(NEW.name,1,1)),
                          LOWER(SUBSTRING(NEW.name FROM 2)));
END//
DELIMITER ;
```

Code as entered:



```
INSERT INTO Player
(p_id, name, weight, speed)
VALUES
('P1', 'MARIO', 6, 6),
('P2', 'Luigi', 6, 6),
('P3', 'Peach', 4, 5),
('P4', 'Daisy', 4, 5),
('P5', 'Yoshi', 4, 5),
('P6', 'Toad', 3, 3),
('P7', 'bOwser', 10, 10),
('P8', 'Wario', 9, 10),
('P9', 'Waluigi', 8, 9),
('P10', 'Mii', 6, 6);
```

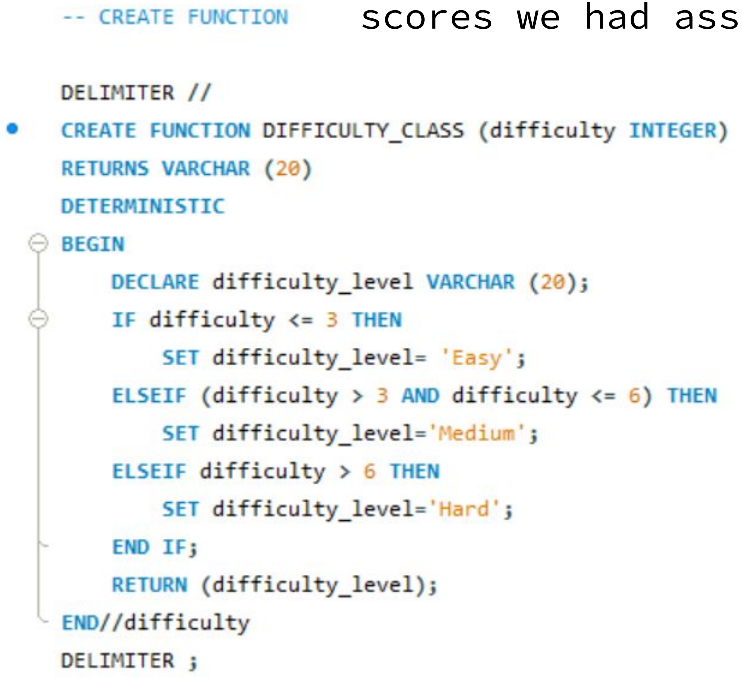
How it displayed as results:



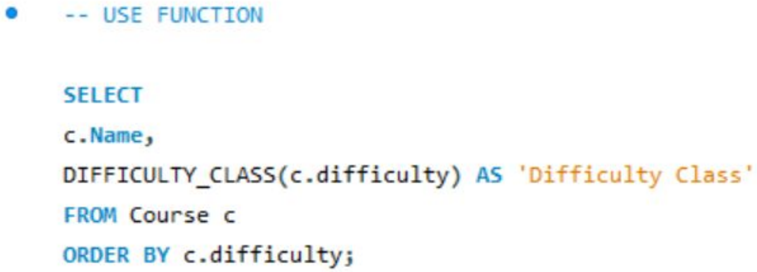
name
Bowser
Daisy
Luigi
Mario
Mii
Peach
Toad
Waluigi
Wario
Yoshi

Function

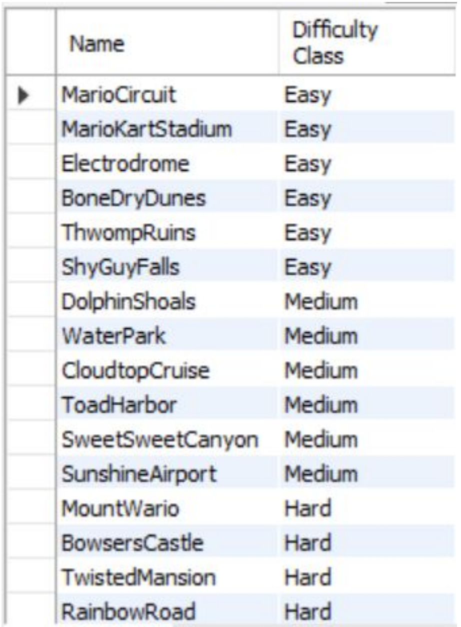
We used a function to label courses as Easy, Medium, or Hard based on the difficulty scores we had assigned them.



```
-- CREATE FUNCTION

DELIMITER //
CREATE FUNCTION DIFFICULTY_CLASS (difficulty INTEGER)
RETURNS VARCHAR (20)
DETERMINISTIC
BEGIN
    DECLARE difficulty_level VARCHAR (20);
    IF difficulty <= 3 THEN
        SET difficulty_level= 'Easy';
    ELSEIF (difficulty > 3 AND difficulty <= 6) THEN
        SET difficulty_level='Medium';
    ELSEIF difficulty > 6 THEN
        SET difficulty_level='Hard';
    END IF;
    RETURN (difficulty_level);
END//difficulty
DELIMITER ;
```



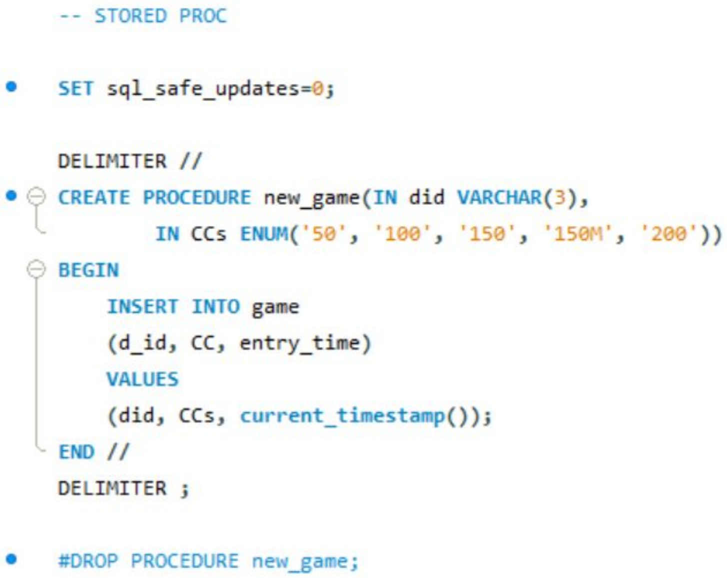
```
-- USE FUNCTION

SELECT
c.Name,
DIFFICULTY_CLASS(c.difficulty) AS 'Difficulty Class'
FROM Course c
ORDER BY c.difficulty;
```



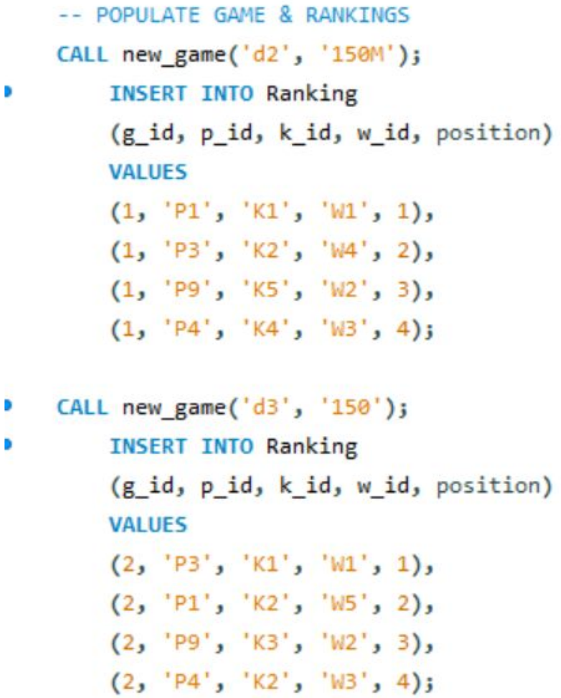
Name	Difficulty Class
MarioCircuit	Easy
MarioKartStadium	Easy
Electrodrome	Easy
BoneDryDunes	Easy
ThwompRuins	Easy
ShyGuyFalls	Easy
DolphinShoals	Medium
WaterPark	Medium
CloudtopCruise	Medium
ToadHarbor	Medium
SweetSweetCanyon	Medium
SunshineAirport	Medium
MountWario	Hard
BowsersCastle	Hard
TwistedMansion	Hard
RainbowRoad	Hard

Stored Procedure

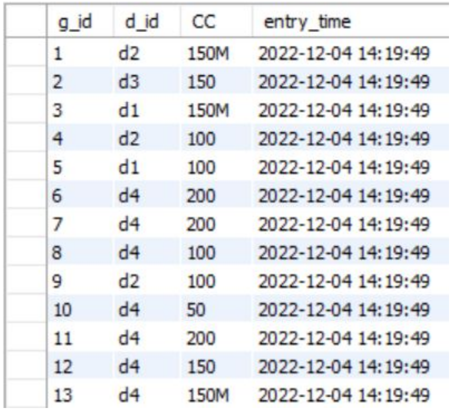
We created a stored procedure to insert new games into the games table with the timestamp of the current time



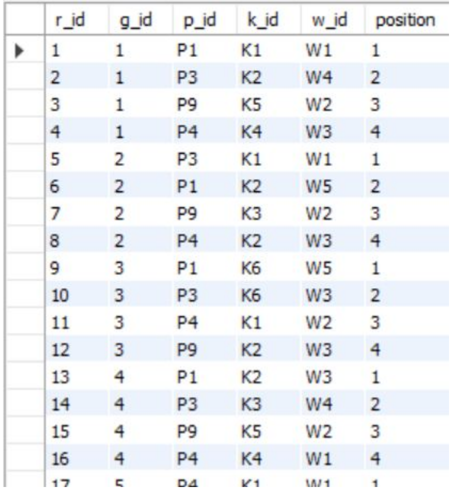
```
-- STORED PROC

SET sql_safe_updates=0;

DELIMITER //
CREATE PROCEDURE new_game(IN did VARCHAR(3),
        IN CCs ENUM('50', '100', '150', '150M', '200'))
BEGIN
    INSERT INTO game
    (d_id, CC, entry_time)
    VALUES
    (did, CCs, current_timestamp());
END //
DELIMITER ;

#DROP PROCEDURE new_game;
```



```
-- POPULATE GAME & RANKINGS
CALL new_game('d2', '150M');
    INSERT INTO Ranking
    (g_id, p_id, k_id, w_id, position)
    VALUES
    (1, 'P1', 'K1', 'W1', 1),
    (1, 'P3', 'K2', 'W4', 2),
    (1, 'P9', 'K5', 'W2', 3),
    (1, 'P4', 'K4', 'W3', 4);

CALL new_game('d3', '150');
    INSERT INTO Ranking
    (g_id, p_id, k_id, w_id, position)
    VALUES
    (2, 'P3', 'K1', 'W1', 1),
    (2, 'P1', 'K2', 'W5', 2),
    (2, 'P9', 'K3', 'W2', 3),
    (2, 'P4', 'K2', 'W3', 4);
```



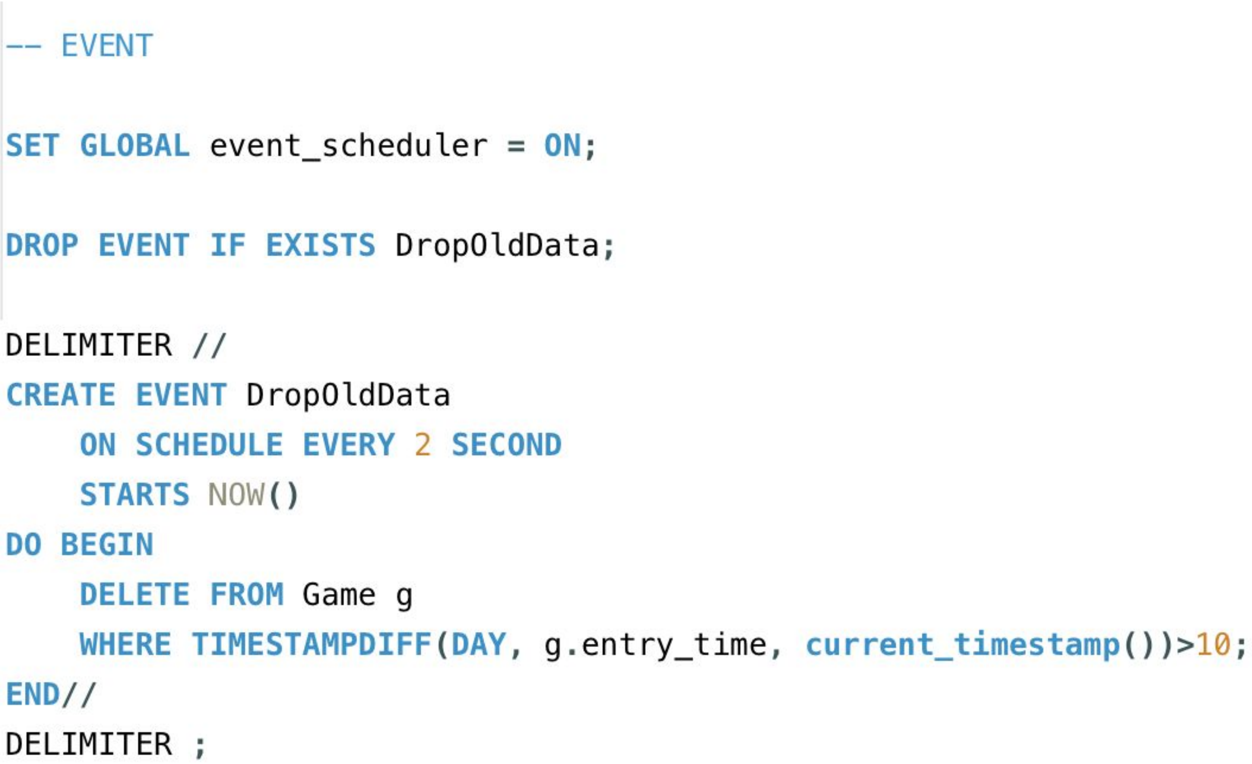
g_id	d_id	CC	entry_time
1	d2	150M	2022-12-04 14:19:49
2	d3	150	2022-12-04 14:19:49
3	d1	150M	2022-12-04 14:19:49
4	d2	100	2022-12-04 14:19:49
5	d1	100	2022-12-04 14:19:49
6	d4	200	2022-12-04 14:19:49
7	d4	200	2022-12-04 14:19:49
8	d4	100	2022-12-04 14:19:49
9	d2	100	2022-12-04 14:19:49
10	d4	50	2022-12-04 14:19:49
11	d4	200	2022-12-04 14:19:49
12	d4	150	2022-12-04 14:19:49
13	d4	150M	2022-12-04 14:19:49



r_id	g_id	p_id	k_id	w_id	position
1	1	P1	K1	W1	1
2	1	P3	K2	W4	2
3	1	P9	K5	W2	3
4	1	P4	K4	W3	4
5	2	P3	K1	W1	1
6	2	P1	K2	W5	2
7	2	P9	K3	W2	3
8	2	P4	K2	W3	4
9	3	P1	K6	W5	1
10	3	P3	K6	W3	2
11	3	P4	K1	W2	3
12	3	P9	K2	W3	4
13	4	P1	K2	W3	1
14	4	P3	K3	W4	2
15	4	P9	K5	W2	3
16	4	P4	K4	W1	4
17	5	P4	K1	W1	1

Event

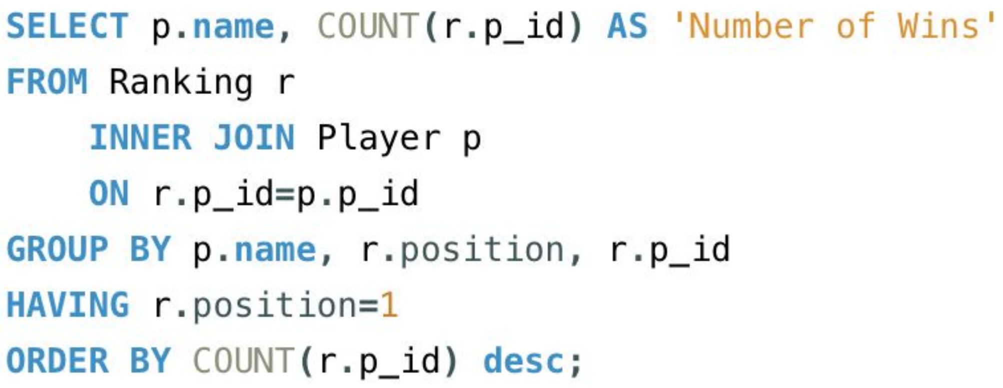
We created an event to get rid of any data older than 10 days.



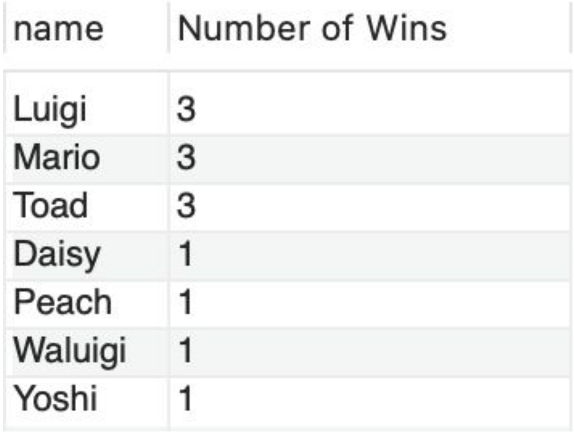
```
-- EVENT

SET GLOBAL event_scheduler = ON;

DROP EVENT IF EXISTS DropOldData;

DELIMITER //
CREATE EVENT DropOldData
    ON SCHEDULE EVERY 2 SECOND
    STARTS NOW()
DO BEGIN
    DELETE FROM Game g
    WHERE TIMESTAMPDIFF(DAY, g.entry_time, current_timestamp())>10;
END//
DELIMITER ;
```

Joins and Group By and Having

We used an INNER JOIN and GROUP BY and HAVING statements to query the number of wins per winning player, ordered by most wins to least.



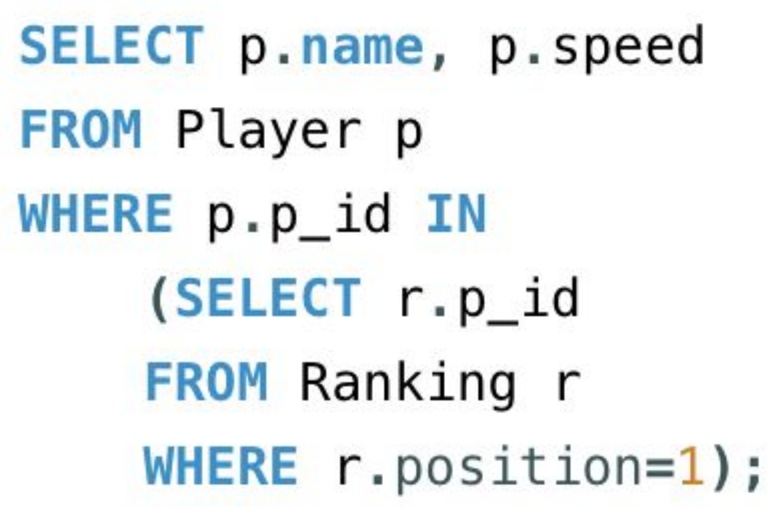
```
SELECT p.name, COUNT(r.p_id) AS 'Number of Wins'
FROM Ranking r
    INNER JOIN Player p
    ON r.p_id=p.p_id
GROUP BY p.name, r.position, r.p_id
HAVING r.position=1
ORDER BY COUNT(r.p_id) desc;
```



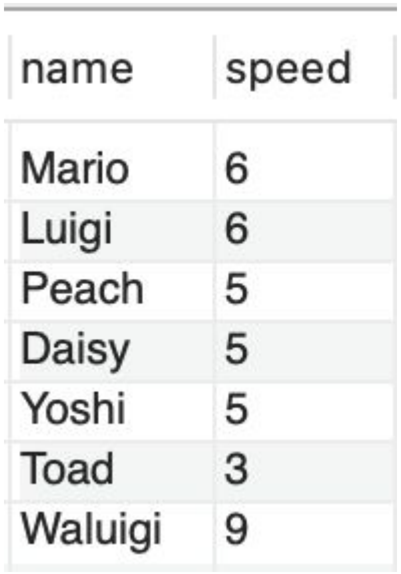
name	Number of Wins
Luigi	3
Mario	3
Toad	3
Daisy	1
Peach	1
Waluigi	1
Yoshi	1

Subquery

We used a query with a subquery to show the player speeds of players who won at least one game.



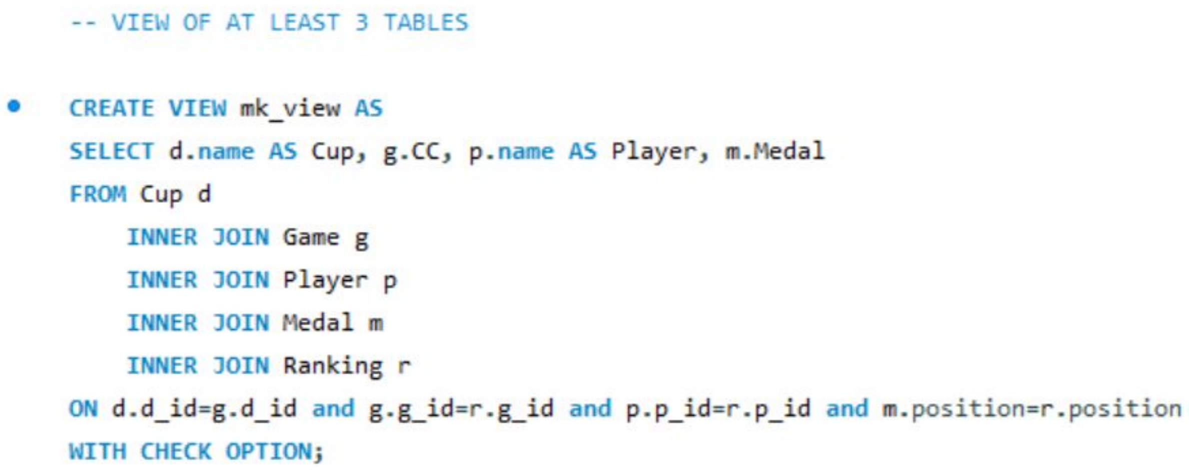
```
SELECT p.name, p.speed
FROM Player p
WHERE p.p_id IN
    (SELECT r.p_id
    FROM Ranking r
    WHERE r.position=1);
```



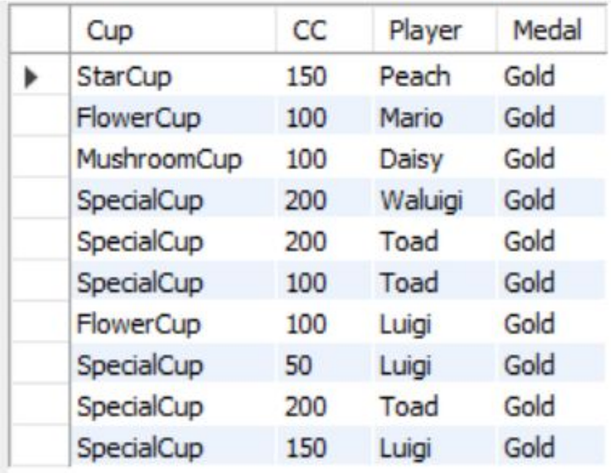
name	speed
Mario	6
Luigi	6
Peach	5
Daisy	5
Yoshi	5
Toad	3
Waluigi	9

Views

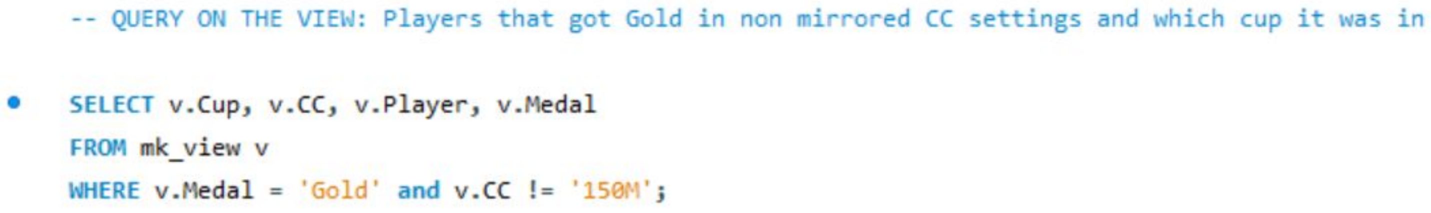
We created a view that only included:
Cup, Player, Medal, CC
Everything else is confidential!



```
-- VIEW OF AT LEAST 3 TABLES

CREATE VIEW mk_view AS
SELECT d.name AS Cup, g.CC, p.name AS Player, m.Medal
FROM Cup d
    INNER JOIN Game g
    INNER JOIN Player p
    INNER JOIN Medal m
    INNER JOIN Ranking r
ON d.d_id=g.d_id and g.g_id=r.g_id and p.p_id=r.p_id and m.position=r.position
WITH CHECK OPTION;
```



	Cup	CC	Player	Medal
▶	StarCup	150	Peach	Gold
	FlowerCup	100	Mario	Gold
	MushroomCup	100	Daisy	Gold
	SpecialCup	200	Waluigi	Gold
	SpecialCup	200	Toad	Gold
	SpecialCup	100	Toad	Gold
	FlowerCup	100	Luigi	Gold
	SpecialCup	50	Luigi	Gold
	SpecialCup	200	Toad	Gold
	SpecialCup	150	Luigi	Gold



```
-- QUERY ON THE VIEW: Players that got Gold in non mirrored CC settings and which cup it was in

SELECT v.Cup, v.CC, v.Player, v.Medal
FROM mk_view v
WHERE v.Medal = 'Gold' and v.CC != '150M';
```

Thanks for listening!