

Final Project

CPS3740 W01: Database Management System

|  |  |
| --- | --- |
| Student Name: | Wang Xinyu |
| Student ID: | 1098648 |
| Lecture: | Dr. Hemn Barzan Abdalla |

Fall 2021-2022

**Database Management System CPS3740**

**Final Project documentation Deadline: 12-14-2021**

* To answers questions below first you needs to create a **schema** which contain at least **four tables**
* Include the constraints like **NOT NULL, UNIQUE and CHECK** constraints in all the table.
* Insert minimum **three** records in each table.
* Retrieve the data using operators **(in, between and like).**
* Minimum five **single row functions** must be applied (**Conversion and date function**).
* **Three queries** must be there from **SET operators (Union, intersect, minus)**
* **Three join queries** must be included out of which **five** must be outer joins.
* **Three nested queries** must be included (Inner Query and Outer Query).
* **Two queries** must be included using **group by and having.**
* **Minimum two views**, which combined of two or three tables, must be included.
* **One cursor and trigger** must be included.
* Create an **application** for your MySQL schema.
* Must have the **main form page**.
* Each table must have (**continues** form with **single** form).
* For two tables should have a **report**.
* The user when open the database must directly show the **main form.**
* Give the summary for your project plus what you learned in this semester, must be less than 300 words.

**Instructions**

* Create a word document which includes details and the screen shots of all the queries as said the above.
* The first page that document contains the information about the student like name, ID Number, with section.
* Use the attached template to prepare your final project document (**Don’t include Normalization).**

# 1. Include the constraints like **NOT NULL, UNIQUE and CHECK** constraints in all the table.

use final\_project;

CREATE TABLE `final\_project`.`country` (

  `country\_id` INT NOT NULL,

  `country\_name` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`country\_id`),

  UNIQUE INDEX `country\_name\_UNIQUE` (`country\_name` ASC) VISIBLE,

  UNIQUE INDEX `country\_id\_UNIQUE` (`country\_id` ASC) VISIBLE);

CREATE TABLE `final\_project`.`element` (

  `element\_id` INT NOT NULL,

  `element\_name` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`element\_id`),

  UNIQUE INDEX `element\_name\_UNIQUE` (`element\_name` ASC) VISIBLE);

CREATE TABLE `final\_project`.`archon` (

  `archon\_id` INT NOT NULL,

  `archon\_name` VARCHAR(45) NOT NULL,

  `archon\_country` VARCHAR(45) NOT NULL,

  `archon\_element` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`archon\_id`),

  FOREIGN KEY (`archon\_country`) REFERENCES `country`(`country\_name`),

  FOREIGN KEY (`archon\_element`) REFERENCES `element`(`element\_name`),

  UNIQUE INDEX `archon\_id` (`archon\_id` ASC) VISIBLE);

CREATE TABLE `final\_project`.`character` (

  `character\_id` INT NOT NULL,

  `character\_name` VARCHAR(45) NOT NULL,

  `character\_age` INT NOT NULL CHECK(`character\_age` >= 5),

  `character\_country` VARCHAR(45) NOT NULL,

  `character\_element` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`character\_id`),

  FOREIGN KEY (`character\_country`) REFERENCES `country`(`country\_name`),

  FOREIGN KEY (`character\_element`) REFERENCES `element`(`element\_name`),

  UNIQUE INDEX `character\_id` (`character\_id` ASC) VISIBLE);

CREATE TABLE `final\_project`.`unpublished\_character` (

  `character\_id` INT NOT NULL,

  `character\_name` VARCHAR(45) NOT NULL,

  `character\_age` INT NOT NULL CHECK(`character\_age` >= 5),

  `character\_country` VARCHAR(45) NOT NULL,

  `character\_element` VARCHAR(45) NOT NULL,

  PRIMARY KEY (`character\_id`),

  FOREIGN KEY (`character\_country`) REFERENCES `country`(`country\_name`),

  FOREIGN KEY (`character\_element`) REFERENCES `element`(`element\_name`),

  UNIQUE INDEX `character\_id` (`character\_id` ASC) VISIBLE);

# 2. Insert minimum **three** records in each table

INSERT INTO `country` (`country\_id`, `country\_name`) VALUES (1, 'Mondstadt');

INSERT INTO `country` (`country\_id`, `country\_name`) VALUES (2, 'Liyue');

INSERT INTO `country` (`country\_id`, `country\_name`) VALUES (3, 'Inadzuma');

INSERT INTO `element` (`element\_id`, `element\_name`) VALUES (1, 'Pyro');

INSERT INTO `element` (`element\_id`, `element\_name`) VALUES (2, 'Geo');

INSERT INTO `element` (`element\_id`, `element\_name`) VALUES (3, 'Dendro');

INSERT INTO `element` (`element\_id`, `element\_name`) VALUES (4, 'Cryo');

INSERT INTO `element` (`element\_id`, `element\_name`) VALUES (5, 'Electro');

INSERT INTO `element` (`element\_id`, `element\_name`) VALUES (6, 'Anemo');

INSERT INTO `element` (`element\_id`, `element\_name`) VALUES (7, 'Hydro');

INSERT INTO `archon` (`archon\_id`, `archon\_name`, `archon\_country`, `archon\_element`) VALUES (1, 'Venti', 'Mondstadt', 'Anemo');

INSERT INTO `archon` (`archon\_id`, `archon\_name`, `archon\_country`, `archon\_element`) VALUES (2, 'Zhongli', 'Liyue', 'Geo');

INSERT INTO `archon` (`archon\_id`, `archon\_name`, `archon\_country`, `archon\_element`) VALUES (3, 'Shougun', 'Inadzuma', 'Electro');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (1, 'Kazuha', 'Inadzuma', 18, 'Anemo');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (2, 'Bennett', 'Mondstadt', 18, 'Geo');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (3, 'Xingqiu', 'Liyue', 18, 'Electro');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (4, 'Ayaka', 'Inadzuma', 18, 'Cryo');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (5, 'Ganyu', 'Liyue', 18, 'Cryo');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (6, 'Albedo', 'Mondstadt', 18, 'Geo');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (7, 'Hutao', 'Liyue', 18, 'Pyro');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (8, 'Xiangling', 'Liyue', 18, 'Pyro');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (9, 'Kokomi', 'Inadzuma', 18, 'Hydro');

INSERT INTO `character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (10, 'Diona', 'Mondstadt', 7, 'Cryo');

INSERT INTO `unpublished\_character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (99, 'Gorou', 'Inadzuma', 18, 'Geo');

INSERT INTO `unpublished\_character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (98, 'Itto', 'Inadzuma', 18, 'Geo');

INSERT INTO `unpublished\_character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (97, 'ShenHe', 'Liyue', 18, 'Cryo');

INSERT INTO `unpublished\_character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (96, 'YunJin', 'Liyue', 18, 'Geo');

*--- suppose this character Diona is unpublished*

INSERT INTO `unpublished\_character` (`character\_id`, `character\_name`, `character\_country`, `character\_age`, `character\_element`) VALUES (95, 'Diona', 'Mondstadt', 7, 'Cryo');

# 3. Retrieve the data using operators **(in, between and like).**

*-- SELECT column\_name(s)*

*-- FROM table\_name*

*-- WHERE column\_name IN (value1, value2, ...);*

SELECT \* FROM final\_project.character

WHERE character\_country IN ('Liyue', 'Inadzuma');

*-- SELECT column\_name(s)*

*-- FROM table\_name*

*-- WHERE column\_name BETWEEN value1 AND value2;*

SELECT \* FROM final\_project.character

WHERE character\_id BETWEEN 2 AND 4;

*-- SELECT column1, column2, ...*

*-- FROM table\_name*

*-- WHERE columnN LIKE pattern;*

SELECT \* FROM final\_project.element

WHERE element\_name LIKE 'A%';

# 4. Minimum five single row functions must be applied (Conversion and date function)

SELECT UPPER(element\_name) FROM final\_project.element;

SELECT CONCAT(UPPER(archon\_name), '\_', UPPER(archon\_country), '\_', UPPER(archon\_element)) FROM final\_project.archon;

SELECT CONCAT(archon\_name, ' , the god of ', LOWER(archon\_element), ' ') FROM final\_project.archon;

SELECT CONCAT(archon\_id, ': ', UPPER(archon\_name)) FROM final\_project.archon;

SELECT CONCAT(character\_name, ': ', character\_element) FROM final\_project.character;

# 5. Three queries must be there from SET operators (Union, intersect, minus)

*-- SET OPERATOR QUERY*

*-- 1. Show all characters inclduing archons in Genshin Impact*

SELECT character\_name FROM final\_project.character

UNION

SELECT archon\_name FROM final\_project.archon;

*-- 2. Show all characters having Cryo Vision and are from Inadzuma*

*-- MySQL does not support INTERSECT operator.*

*-- SELECT character\_name FROM final\_project.character WHERE character\_element LIKE "Cryo"*

*-- INTERSECT*

*-- SELECT character\_name FROM final\_project.character WHERE character\_country IS "Inadzuma";*

*-- 3. Show all characters having Cryo Vision and are not from Inadzuma*

*-- MySQL does not support MINUS operator*

*-- SELECT character\_name FROM final\_project.character WHERE character\_element LIKE "Cryo"*

*-- MINUS*

*-- SELECT character\_name FROM final\_project.character WHERE character\_country Like "Inadzuma";*

*-- 4. Show characters and Mondstadt and the archon of "Inadzuma"*

SELECT character\_name FROM final\_project.character WHERE character\_country = "Mondstadt"

UNION

SELECT archon\_name FROM final\_project.archon WHERE archon\_country = "Inadzuma";

*-- 5. Show all children character and Anemo archon*

SELECT character\_name FROM final\_project.character WHERE character\_age < 18

UNION

SELECT archon\_name FROM final\_project.archon WHERE archon\_element = "Anemo"

*-- JOIN QUERY*

SELECT archon\_name FROM final\_project.archon

FULL OUTER JOIN final\_project.character

ON final\_project.archon.archon\_element = final\_project.character.character\_element

# 6. **Three join queries** must be included out of which **five** must be outer joins

use final\_project;

SELECT `character`.character\_id, unpublished\_character.character\_element, unpublished\_character.character\_name

FROM unpublished\_character

LEFT JOIN `character`

ON `character`.character\_name = unpublished\_character.character\_name;

SELECT `character`.character\_id, unpublished\_character.character\_element, unpublished\_character.character\_name

FROM unpublished\_character

RIGHT JOIN `character`

ON `character`.character\_name = unpublished\_character.character\_name;

SELECT `character`.character\_id, unpublished\_character.character\_element, unpublished\_character.character\_name

FROM unpublished\_character

LEFT JOIN `character`

ON `character`.character\_name = unpublished\_character.character\_name

WHERE unpublished\_character.character\_element = 'Cryo';

# 7. **Three nested queries** must be included (Inner Query and Outer Query).

use final\_project;

SELECT character\_name FROM `character` WHERE character\_country IN (SELECT character\_country FROM unpublished\_character);

SELECT character\_id, character\_name FROM `character` WHERE character\_age IN (SELECT character\_age FROM unpublished\_character);

SELECT character\_id, character\_name FROM `character` WHERE character\_element IN (SELECT archon\_element FROM archon);

# 8. Two queries must be included using group by and having

use final\_project;

SELECT character\_element, COUNT(character\_element) AS TOTALNUMBER FROM `character` GROUP BY character\_element HAVING TOTALNUMBER > 1;

SELECT character\_country, COUNT(character\_country) AS TOTALNUMBER FROM `character` GROUP BY character\_country HAVING TOTALNUMBER > 3;

# 9. **Minimum two views**, which combined of two or three tables, must be included

use final\_project;

CREATE VIEW ALL\_CHARACTER\_NAME AS

SELECT archon\_name FROM archon

UNION

SELECT character\_name FROM `character`;

CREATE VIEW ALL\_CRYO\_CHARACTER\_NAME AS

SELECT archon\_name FROM archon WHERE archon\_element = 'Cryo'

UNION

SELECT character\_name FROM `character` WHERE character\_element = 'Cryo';

# 10. One cursor and trigger must be included

Cursor

DELIMITER $$

CREATE PROCEDURE createCharacterList (

    INOUT characterList varchar(4000)

)

BEGIN

    DECLARE finished INTEGER DEFAULT 0;

    DECLARE mycharacter varchar(100) DEFAULT "";

*-- declare cursor for character*

    DEClARE curChracter

        CURSOR FOR

            SELECT character\_name FROM final\_project.character;

*-- declare NOT FOUND handler*

    DECLARE CONTINUE HANDLER

        FOR NOT FOUND SET finished = 1;

    OPEN curChracter;

    getCharacter: LOOP

        FETCH curChracter INTO mycharacter;

        IF finished = 1 THEN

            LEAVE getCharacter;

        END IF;

*-- build character list*

        SET characterList = CONCAT(mycharacter,";",characterList);

    END LOOP getCharacter;

    CLOSE curChracter;

END$$

DELIMITER ;

SET @characterList = "";

CALL createCharacterList(@characterList);

SELECT @characterList;

Trigger

CREATE TRIGGER AUTO\_UPDATE\_COUNTRY

AFTER UPDATE ON final\_project.country

FOR EACH ROW

    UPDATE final\_project.character SET character\_country = NEW.country\_name

    WHERE character\_country = OLD.country\_name;

# 11. Create an **application** for your MySQL schema.

Graphical user interface, application

Description automatically generated

## Must have the **main form page**.

The main form page of access application is shown below:

Graphical user interface

Description automatically generated

## Each table must have (**continues** form with **single** form).

The continues form of Archon Table

Table

Description automatically generated

The continues form of Character Table

Table

Description automatically generated

The continues form of Country Table

Table

Description automatically generated

## For two tables should have a **report**.

The report form of Archon Table

Graphical user interface, application

Description automatically generated

The report form of Character Table

Table

Description automatically generated

## The user when open the database must directly show the **main form.**

The main form is always the firstly form when the user opens the database

Graphical user interface, application

Description automatically generated

# 12. Give the summary for your project plus what you learned in this semester, must be less than 300 words.

In this project, I used part of the data from the game Genshin Impact and processed it using SQL (MySQL). For example, with the SQL statement Group By, I can visually organize and sort the characters in the game Genshin Impact from different countries such as Liyue, Mondstadt, Inadzuma, without having to write the functional modules about them while programming. And with the trigger, the database could automatically update the record, for instance, when I update the name of a country, the trigger will be activated and perform the code I wrote to update the other record that contains the old country name and replace them with the new name, such that I won’t worry about that.

Through this semester's study of databases, I have gained a deeper understanding of the role and significance of databases. Like in software development, the concepts of Model-View-Controller (aka. MVC) or Model-View-ViewModel (aka. MVVM) allow developers to minimize bugs and facilitate debugging by separating the different functions of software development. Using a separate database, rather than creating a new database and rebuilding a new set of database logic in the program, will increases code reuse just as no one writes large programs in assembly language today, so it could greatly improve development efficiency, and since the database is not tightly coupled to the main program, future programs can easily update the database, thus facilitating future software updates.