DBS Final project-PostgreSQL

玩家隊伍總傷害、角色統計資料庫

姓名：羅海綺、林雨臻

學號：11127132、11127138

班級：資訊二甲

壹、前言：

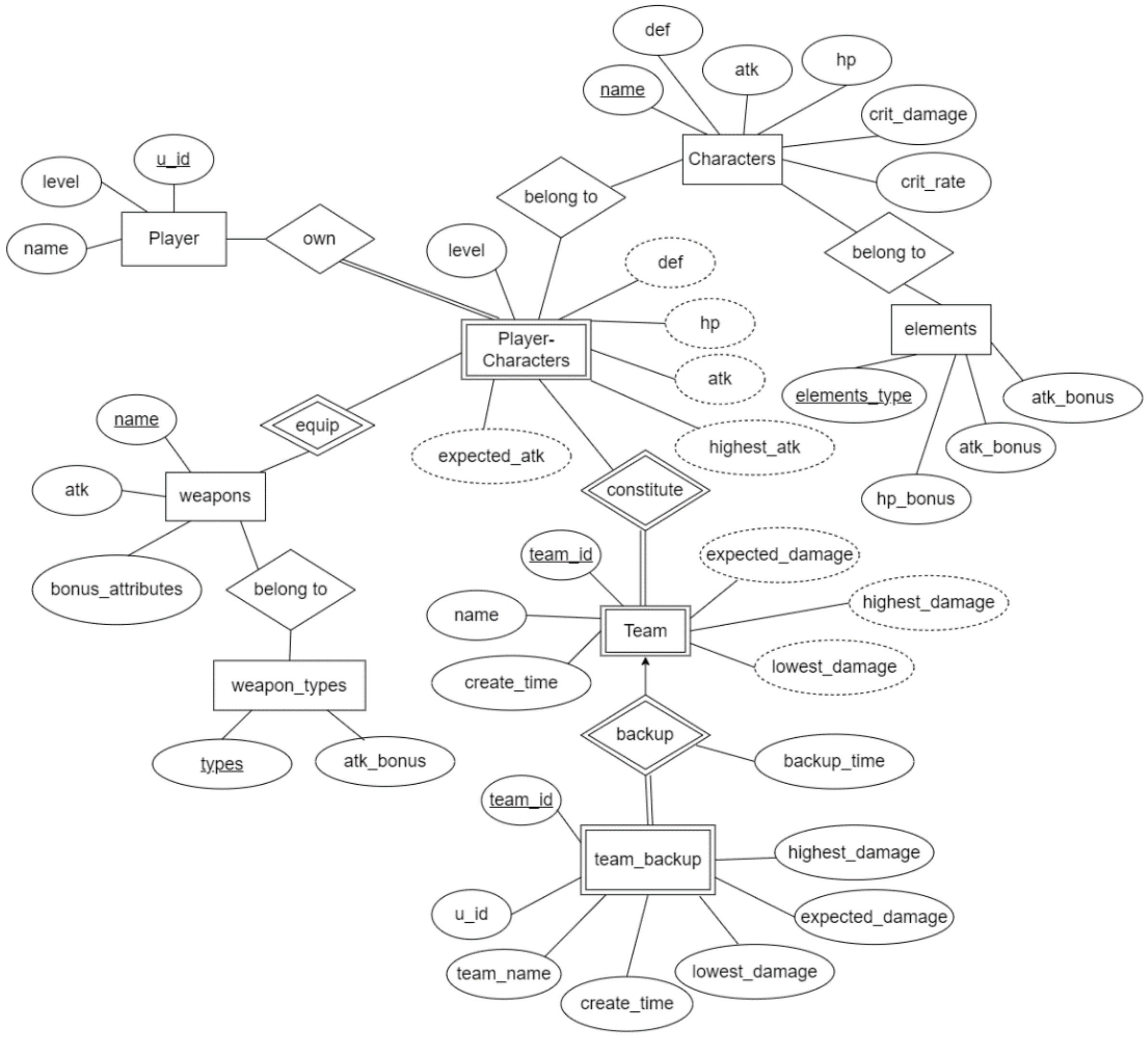
學習資料庫一個學期，從資料庫系統概論、relational data model、ER diagram 到 SQL，我們對於資料庫系統有更深入的了解，也學習如何應用、創建資料庫。期末專題要我們運用學到的內容從 ER diagram 開始設計，創建出一個資料庫。

貳、內文：

一、設計簡介：

基於名為原神的遊戲而設計的資料庫，提供玩家查詢資料、計算傷害、回溯數據、提升建議等等功能。

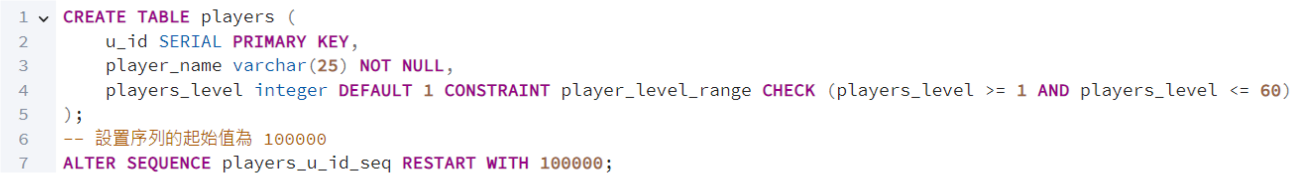
二、ER diagram：



三、SQL code：

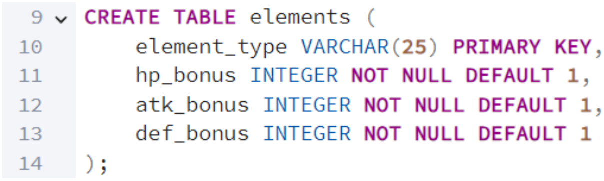
1. TABLE：

(a) players:



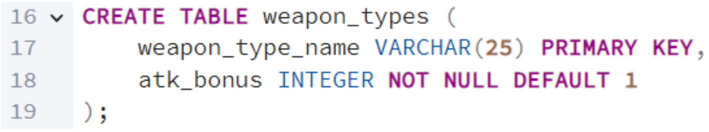
```sql
CREATE TABLE players (
    u_id SERIAL PRIMARY KEY,
    player_name varchar(25) NOT NULL,
    players_level integer DEFAULT 1 CONSTRAINT player_level_range CHECK (players_level >= 1 AND players_level <= 60)
);
-- 設置序列的起始值為 100000
ALTER SEQUENCE players_u_id_seq RESTART WITH 100000;
```

(b) elements:



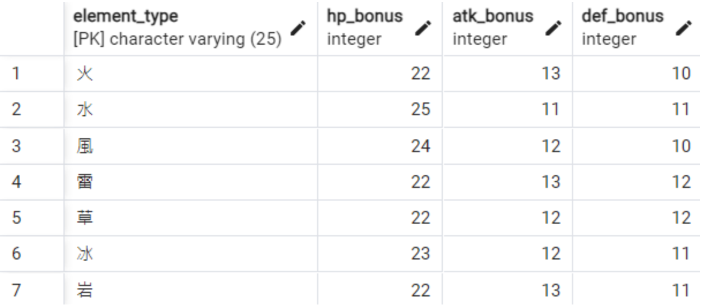
```sql
CREATE TABLE elements (
    element_type VARCHAR(25) PRIMARY KEY,
    hp_bonus INTEGER NOT NULL DEFAULT 1,
    atk_bonus INTEGER NOT NULL DEFAULT 1,
    def_bonus INTEGER NOT NULL DEFAULT 1
);
```

(c) weapon_types:



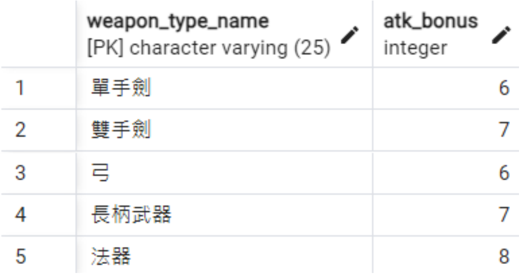
```sql
CREATE TABLE weapon_types (
    weapon_type_name VARCHAR(25) PRIMARY KEY,
    atk_bonus INTEGER NOT NULL DEFAULT 1
);
```

elements data:



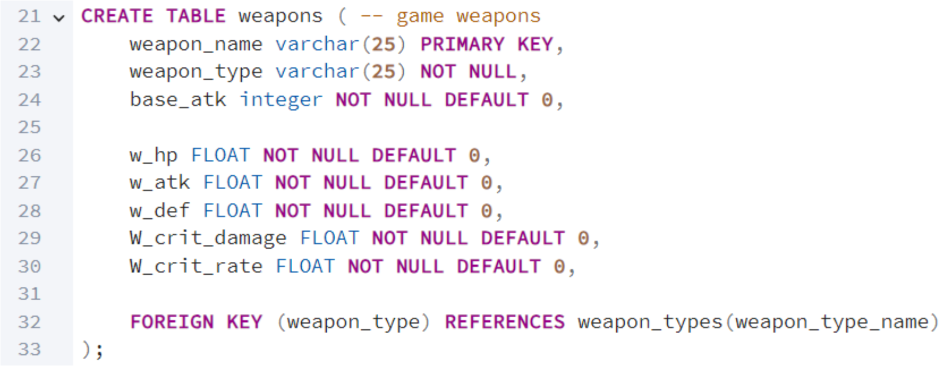
	element_type [PK] character varying (25)	hp_bonus integer	atk_bonus integer	def_bonus integer
1	火	22	13	10
2	水	25	11	11
3	風	24	12	10
4	雷	22	13	12
5	草	22	12	12
6	冰	23	12	11
7	岩	22	13	11

weapon_types data:



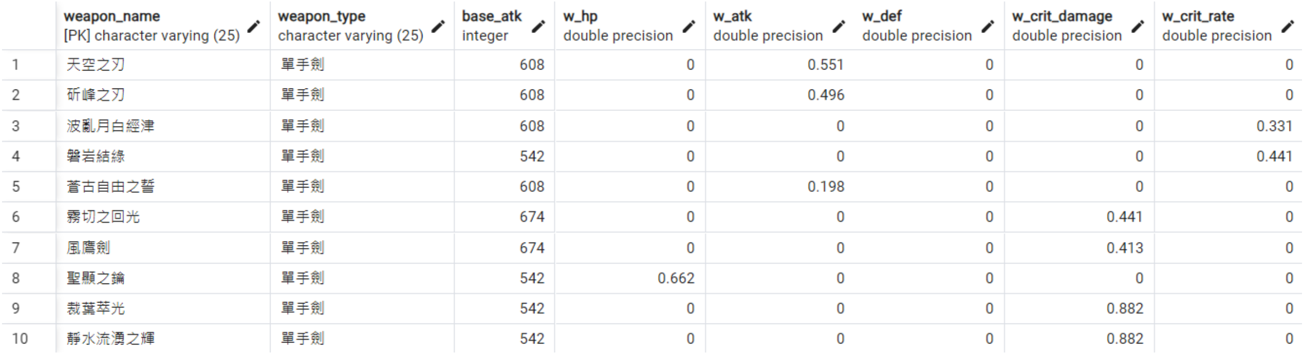
	weapon_type_name [PK] character varying (25)	atk_bonus integer
1	單手劍	6
2	雙手劍	7
3	弓	6
4	長柄武器	7
5	法器	8

(d) weapons:



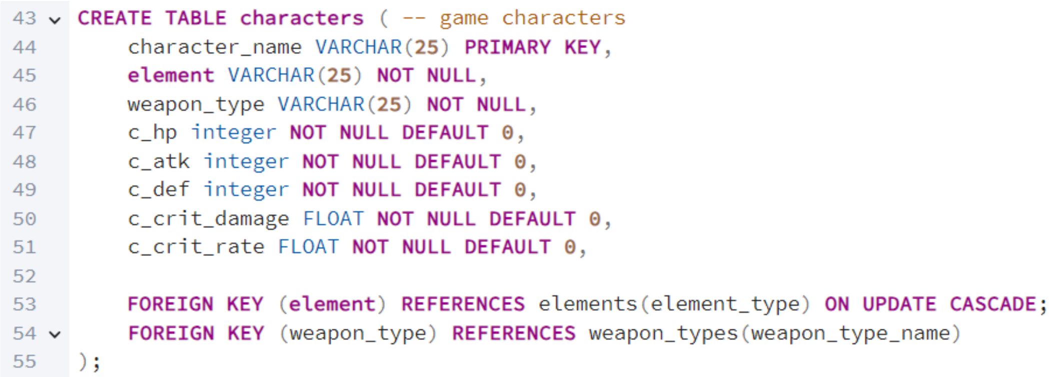
```sql
CREATE TABLE weapons ( -- game weapons
    weapon_name varchar(25) PRIMARY KEY,
    weapon_type varchar(25) NOT NULL,
    base_atk integer NOT NULL DEFAULT 0,

    w_hp FLOAT NOT NULL DEFAULT 0,
    w_atk FLOAT NOT NULL DEFAULT 0,
    w_def FLOAT NOT NULL DEFAULT 0,
    W_crit_damage FLOAT NOT NULL DEFAULT 0,
    W_crit_rate FLOAT NOT NULL DEFAULT 0,

    FOREIGN KEY (weapon_type) REFERENCES weapon_types(weapon_type_name)
);
```

(部分資料)



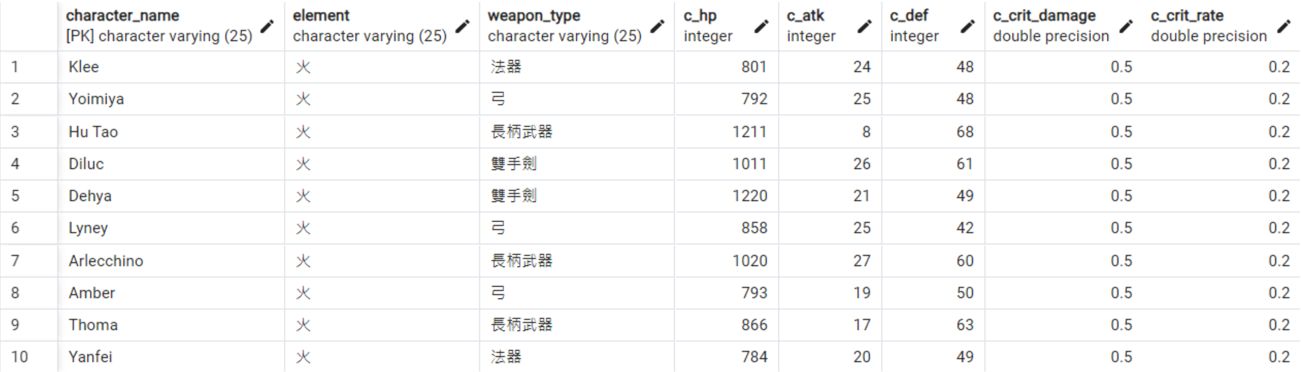
	weapon_name [PK] character varying (25)	weapon_type character varying (25)	base_atk integer	w_hp double precision	w_atk double precision	w_def double precision	w_crit_damage double precision	w_crit_rate double precision
1	天空之刃	單手劍	608	0	0.551	0	0	0
2	斫峰之刃	單手劍	608	0	0.496	0	0	0
3	波亂月白經津	單手劍	608	0	0	0	0	0.331
4	磐岩結綠	單手劍	542	0	0	0	0	0.441
5	蒼古自由之誓	單手劍	608	0	0.198	0	0	0
6	霧切之回光	單手劍	674	0	0	0	0.441	0
7	風鷹劍	單手劍	674	0	0	0	0.413	0
8	聖顯之鑰	單手劍	542	0.662	0	0	0	0
9	裁葉萃光	單手劍	542	0	0	0	0.882	0
10	靜水流湧之輝	單手劍	542	0	0	0	0.882	0

(e) characters:



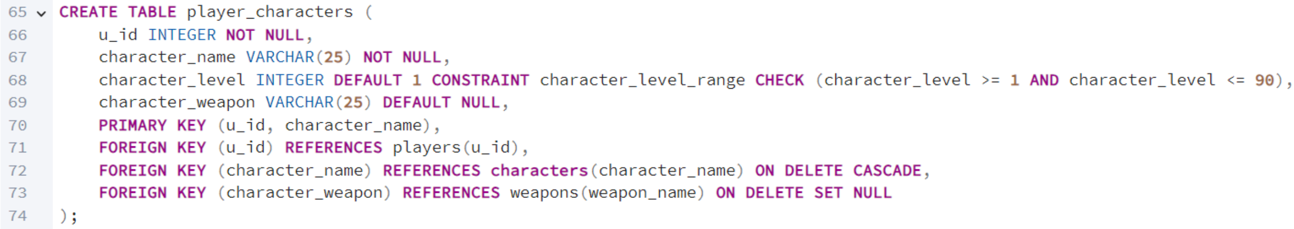
```
CREATE TABLE characters ( -- game characters
    character_name VARCHAR(25) PRIMARY KEY,
    element VARCHAR(25) NOT NULL,
    weapon_type VARCHAR(25) NOT NULL,
    c_hp integer NOT NULL DEFAULT 0,
    c_atk integer NOT NULL DEFAULT 0,
    c_def integer NOT NULL DEFAULT 0,
    c_crit_damage FLOAT NOT NULL DEFAULT 0,
    c_crit_rate FLOAT NOT NULL DEFAULT 0,

    FOREIGN KEY (element) REFERENCES elements(element_type) ON UPDATE CASCADE;
    FOREIGN KEY (weapon_type) REFERENCES weapon_types(weapon_type_name)
);
```

(部分資料)



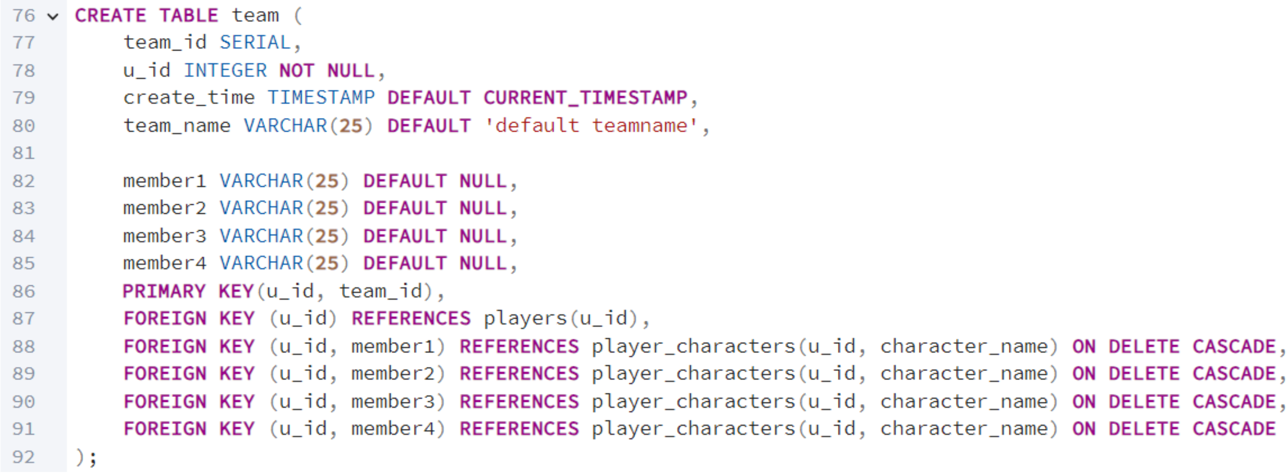
	character_name [PK] character varying (25)	element character varying (25)	weapon_type character varying (25)	c_hp integer	c_atk integer	c_def integer	c_crit_damage double precision	c_crit_rate double precision
1	Klee	火	法器	801	24	48	0.5	0.2
2	Yoimiya	火	弓	792	25	48	0.5	0.2
3	Hu Tao	火	長柄武器	1211	8	68	0.5	0.2
4	Diluc	火	雙手劍	1011	26	61	0.5	0.2
5	Dehya	火	雙手劍	1220	21	49	0.5	0.2
6	Lyney	火	弓	858	25	42	0.5	0.2
7	Arlecchino	火	長柄武器	1020	27	60	0.5	0.2
8	Amber	火	弓	793	19	50	0.5	0.2
9	Thoma	火	長柄武器	866	17	63	0.5	0.2
10	Yanfei	火	法器	784	20	49	0.5	0.2

(f) player_characters:



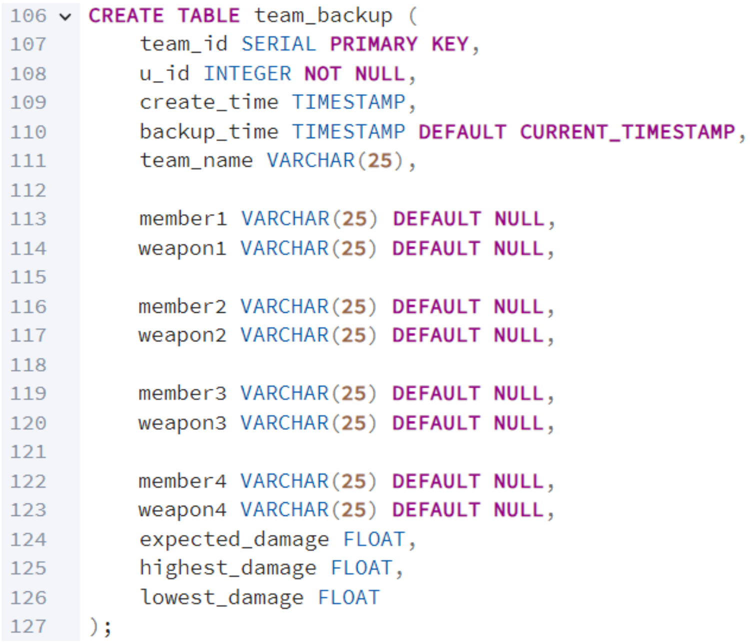
```
CREATE TABLE player_characters (
    u_id INTEGER NOT NULL,
    character_name VARCHAR(25) NOT NULL,
    character_level INTEGER DEFAULT 1 CONSTRAINT character_level_range CHECK (character_level >= 1 AND character_level <= 90),
    character_weapon VARCHAR(25) DEFAULT NULL,
    PRIMARY KEY (u_id, character_name),
    FOREIGN KEY (u_id) REFERENCES players(u_id),
    FOREIGN KEY (character_name) REFERENCES characters(character_name) ON DELETE CASCADE,
    FOREIGN KEY (character_weapon) REFERENCES weapons(weapon_name) ON DELETE SET NULL
);
```

(g) team:



```
CREATE TABLE team (
    team_id SERIAL,
    u_id INTEGER NOT NULL,
    create_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    team_name VARCHAR(25) DEFAULT 'default teamname',

    member1 VARCHAR(25) DEFAULT NULL,
    member2 VARCHAR(25) DEFAULT NULL,
    member3 VARCHAR(25) DEFAULT NULL,
    member4 VARCHAR(25) DEFAULT NULL,
    PRIMARY KEY(u_id, team_id),
    FOREIGN KEY (u_id) REFERENCES players(u_id),
    FOREIGN KEY (u_id, member1) REFERENCES player_characters(u_id, character_name) ON DELETE CASCADE,
    FOREIGN KEY (u_id, member2) REFERENCES player_characters(u_id, character_name) ON DELETE CASCADE,
    FOREIGN KEY (u_id, member3) REFERENCES player_characters(u_id, character_name) ON DELETE CASCADE,
    FOREIGN KEY (u_id, member4) REFERENCES player_characters(u_id, character_name) ON DELETE CASCADE
);
```

(h) team_backup:



```sql
CREATE TABLE team_backup (
    team_id SERIAL PRIMARY KEY,
    u_id INTEGER NOT NULL,
    create_time TIMESTAMP,
    backup_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    team_name VARCHAR(25),

    member1 VARCHAR(25) DEFAULT NULL,
    weapon1 VARCHAR(25) DEFAULT NULL,

    member2 VARCHAR(25) DEFAULT NULL,
    weapon2 VARCHAR(25) DEFAULT NULL,

    member3 VARCHAR(25) DEFAULT NULL,
    weapon3 VARCHAR(25) DEFAULT NULL,

    member4 VARCHAR(25) DEFAULT NULL,
    weapon4 VARCHAR(25) DEFAULT NULL,
    expected_damage FLOAT,
    highest_damage FLOAT,
    lowest_damage FLOAT
);
```

2. Trigger：

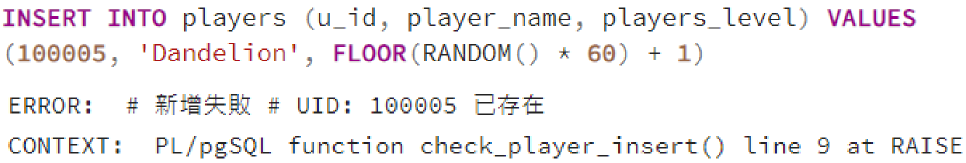
(a) player_insert_trigger：

(i) 觸發條件：before insert player

(ii) Trigger function：check_player_insert()

(iii) 目的：

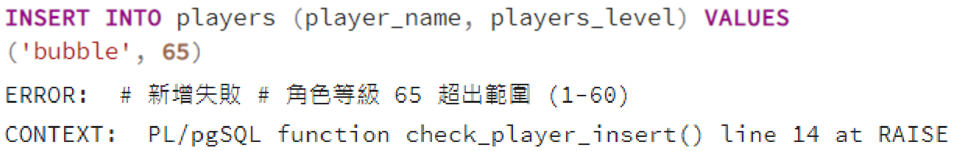
- 檢查是否有相同 UID 存在



```sql
INSERT INTO players (u_id, player_name, players_level) VALUES
(100005, 'Dandelion', FLOOR(RANDOM() * 60) + 1)
```

```
ERROR:  # 新增失敗 # UID: 100005 已存在
CONTEXT:  PL/pgSQL function check_player_insert() line 9 at RAISE
```

- 檢查玩家等級範圍



```sql
INSERT INTO players (player_name, players_level) VALUES
('bubble', 65)
```

```
ERROR:  # 新增失敗 # 角色等級 65 超出範圍 (1-60)
CONTEXT:  PL/pgSQL function check_player_insert() line 14 at RAISE
```

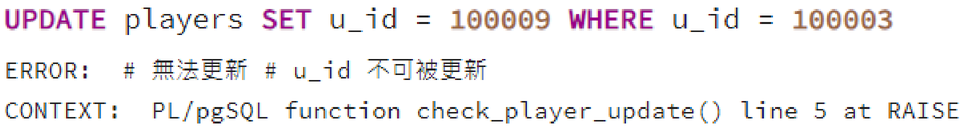
(b) player_update_trigger：

(i) 觸發條件：before update player

(ii) Trigger function：check_player_update()

(iii) 目的：

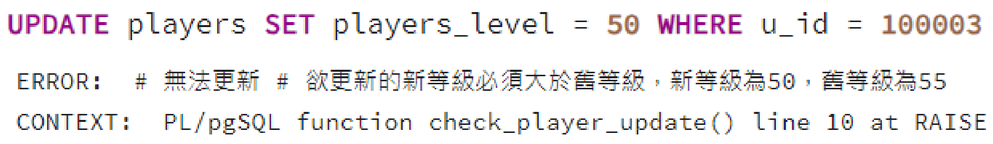
- 確保 UID 無法被更新



```sql
UPDATE players SET u_id = 100009 WHERE u_id = 100003
```

```
ERROR:  # 無法更新 # u_id 不可被更新
CONTEXT:  PL/pgSQL function check_player_update() line 5 at RAISE
```

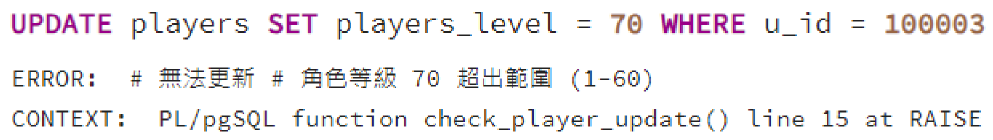
- 限制更新玩家等級時，更新等級不能小於舊等級



```
UPDATE players SET players_level = 50 WHERE u_id = 100003
```

```
ERROR:  # 無法更新 # 欲更新的新等級必須大於舊等級，新等級為50，舊等級為55
CONTEXT:  PL/pgSQL function check_player_update() line 10 at RAISE
```

- 檢查玩家等級範圍



```
UPDATE players SET players_level = 70 WHERE u_id = 100003
```

```
ERROR:  # 無法更新 # 角色等級 70 超出範圍 (1-60)
CONTEXT:  PL/pgSQL function check_player_update() line 15 at RAISE
```

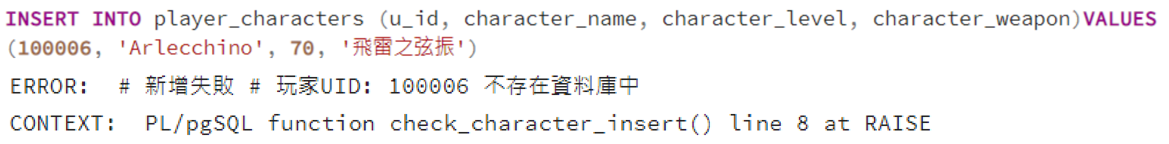
(c) character_insert_trigger：

(i) 觸發條件：before insert player_ characters

(ii) Trigger function：check_character_insert()

(iii) 目的：

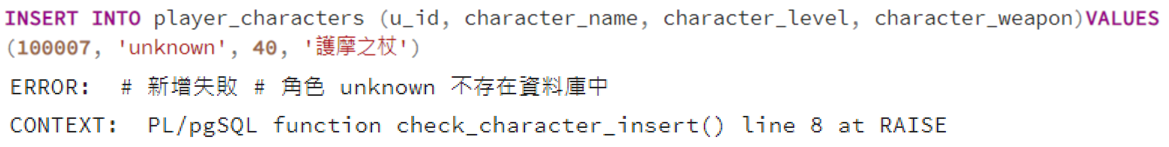
- 檢查 UID 是否存在於 players 表中



```
INSERT INTO player_characters (u_id, character_name, character_level, character_weapon)VALUES
(100006, 'Arlecchino', 70, '飛雷之弦振')
```

```
ERROR:  # 新增失敗 # 玩家UID: 100006 不存在資料庫中
CONTEXT:  PL/pgSQL function check_character_insert() line 8 at RAISE
```

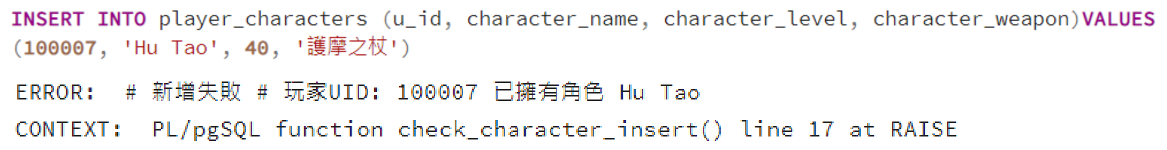
- 檢查角色是否存在 characters 表中



```
INSERT INTO player_characters (u_id, character_name, character_level, character_weapon)VALUES
(100007, 'unknown', 40, '護摩之杖')
```

```
ERROR:  # 新增失敗 # 角色 unknown 不存在資料庫中
CONTEXT:  PL/pgSQL function check_character_insert() line 8 at RAISE
```

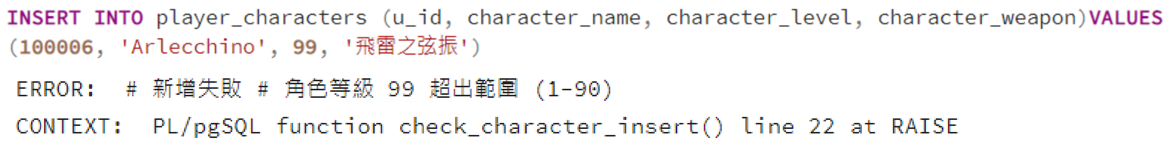
- 檢查該角色玩家是否已擁有



```
INSERT INTO player_characters (u_id, character_name, character_level, character_weapon)VALUES
(100007, 'Hu Tao', 40, '護摩之杖')
```

```
ERROR:  # 新增失敗 # 玩家UID: 100007 已擁有角色 Hu Tao
CONTEXT:  PL/pgSQL function check_character_insert() line 17 at RAISE
```

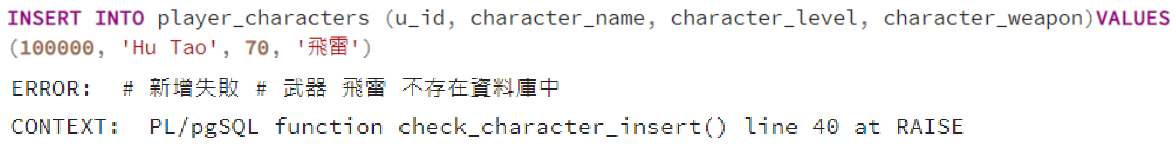
- 檢查角色等級範圍



```
INSERT INTO player_characters (u_id, character_name, character_level, character_weapon)VALUES
(100006, 'Arlecchino', 99, '飛雷之弦振')
```

```
ERROR:  # 新增失敗 # 角色等級 99 超出範圍 (1-90)
CONTEXT:  PL/pgSQL function check_character_insert() line 22 at RAISE
```

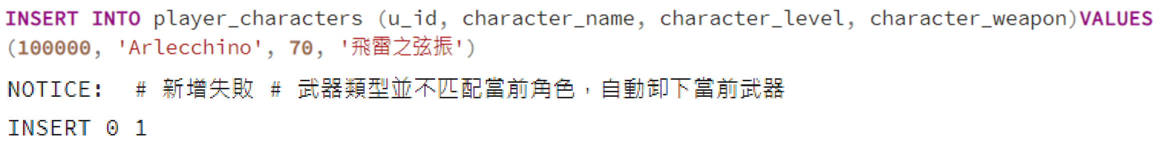
- 檢查武器是否存在 weapons 表中



```
INSERT INTO player_characters (u_id, character_name, character_level, character_weapon)VALUES
(100000, 'Hu Tao', 70, '飛雷')
```

```
ERROR:  # 新增失敗 # 武器 飛雷 不存在資料庫中
CONTEXT:  PL/pgSQL function check_character_insert() line 40 at RAISE
```

- 確認武器與角色是否匹配



```
INSERT INTO player_characters (u_id, character_name, character_level, character_weapon)VALUES
(100000, 'Arlecchino', 70, '飛雷之弦振')
```

```
NOTICE:  # 新增失敗 # 武器類型並不匹配當前角色，自動卸下當前武器
INSERT 0 1
```

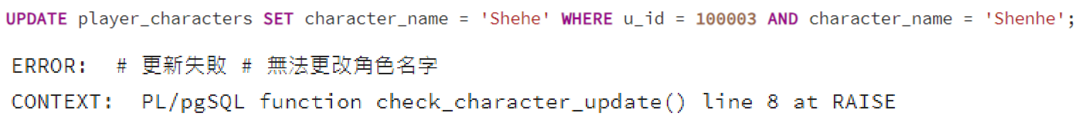
(d) character_update_trigger：

(i) 觸發條件：before update player_ characters

(ii) Trigger function：check_character_update()

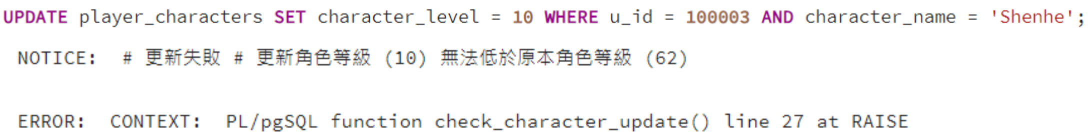
(iii) 目的：

- 檢查玩家是否要更改角色名字



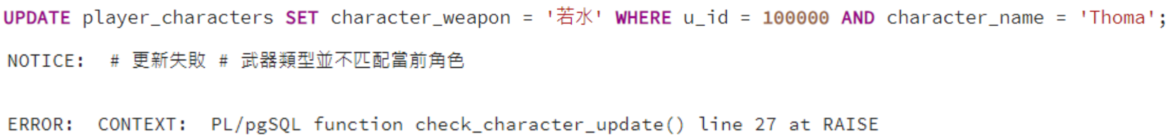
```
UPDATE player_characters SET character_name = 'Shehe' WHERE u_id = 100003 AND character_name = 'Shenhe';

ERROR:  # 更新失敗 # 無法更改角色名字
CONTEXT:  PL/pgSQL function check_character_update() line 8 at RAISE
```

- 檢查角色等級是否下降



```
UPDATE player_characters SET character_level = 10 WHERE u_id = 100003 AND character_name = 'Shenhe';

NOTICE:  # 更新失敗 # 更新角色等級 (10) 無法低於原本角色等級 (62)

ERROR:  CONTEXT:  PL/pgSQL function check_character_update() line 27 at RAISE
```

- 確認武器與角色是否匹配



```
UPDATE player_characters SET character_weapon = '若水' WHERE u_id = 100000 AND character_name = 'Thoma';

NOTICE:  # 更新失敗 # 武器類型並不匹配當前角色

ERROR:  CONTEXT:  PL/pgSQL function check_character_update() line 27 at RAISE
```

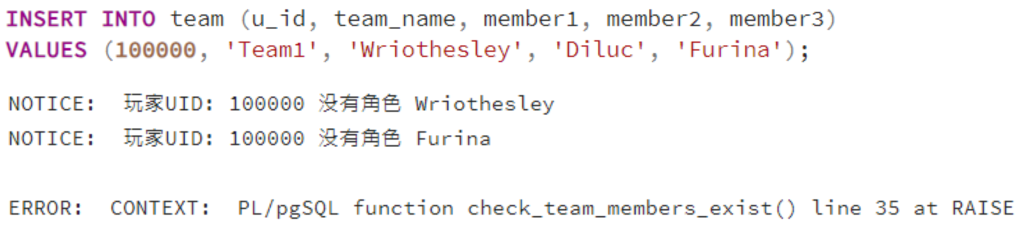
(e) before_insert_team：

(i) 觸發條件：before insert team

(ii) Trigger function：check_team_members_exist()

(iii) 目的：

- 檢查每個隊伍成員是否存在於玩家的角色列表資料中



```
INSERT INTO team (u_id, team_name, member1, member2, member3)
VALUES (100000, 'Team1', 'Wriothesley', 'Diluc', 'Furina');

NOTICE:  玩家UID: 100000 沒有角色 Wriothesley
NOTICE:  玩家UID: 100000 沒有角色 Furina

ERROR:  CONTEXT:  PL/pgSQL function check_team_members_exist() line 35 at RAISE
```

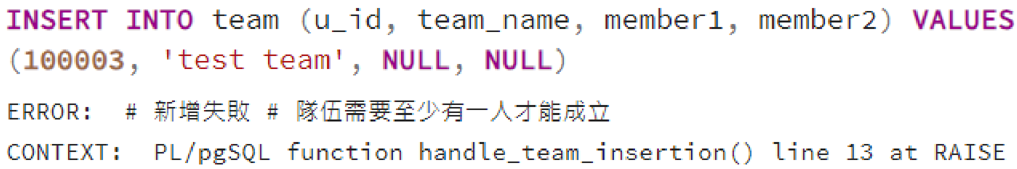
(f) trigger_handle_team_insertion：

(i) 觸發條件：before insert team

(ii) Trigger function：handle_team_insertion()

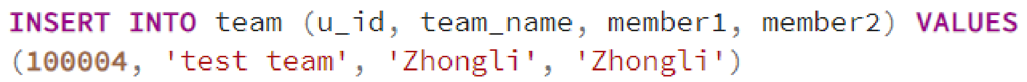
(iii) 目的：

- 確保至少有一個成員存在

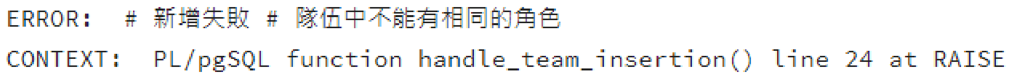


```
INSERT INTO team (u_id, team_name, member1, member2) VALUES
(100003, 'test team', NULL, NULL)

ERROR:  # 新增失敗 # 隊伍需要至少有一人才能成立
CONTEXT:  PL/pgSQL function handle_team_insertion() line 13 at RAISE
```

- 確保隊伍中沒有相同的角色

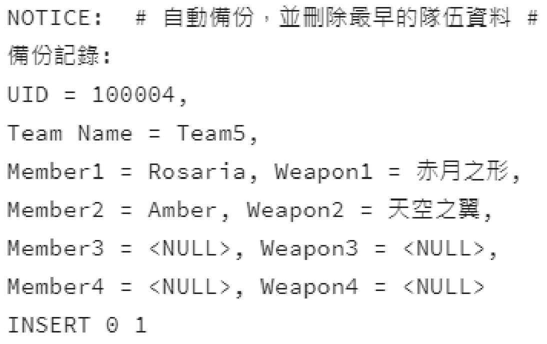


```
INSERT INTO team (u_id, team_name, member1, member2) VALUES
(100004, 'test team', 'Zhongli', 'Zhongli')
```



```
ERROR:  # 新增失敗 # 隊伍中不能有相同的角色
CONTEXT:  PL/pgSQL function handle_team_insertion() line 24 at RAISE
```

- 判斷是否要將隊伍資料備份，若要則自動備份



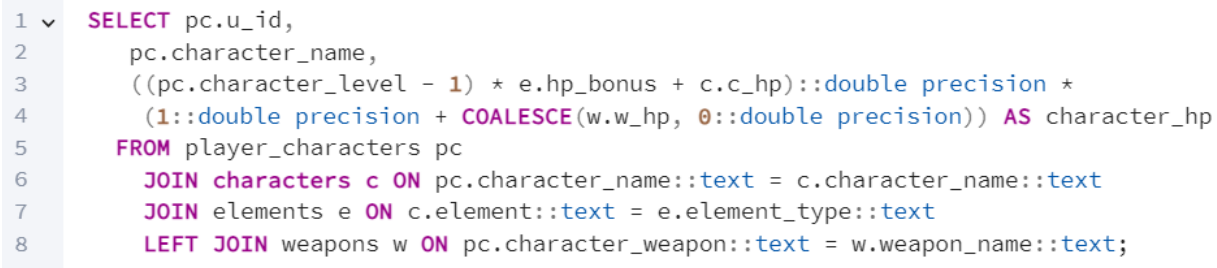
```
NOTICE:  # 自動備份，並刪除最早的隊伍資料 #
備份記錄:
UID = 100004,
Team Name = Team5,
Member1 = Rosaria, Weapon1 = 赤月之形,
Member2 = Amber, Weapon2 = 天空之翼,
Member3 = <NULL>, Weapon3 = <NULL>,
Member4 = <NULL>, Weapon4 = <NULL>
INSERT 0 1
```

3. View：

(a) view_character_hp：

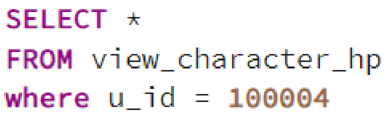
依照公式：[(角色等級-1) * 角色屬性的生命值每等級增加數值+角色等級為 1 時的基礎生命值] * (1+裝備武器的生命值加成%)

計算角色生命值。



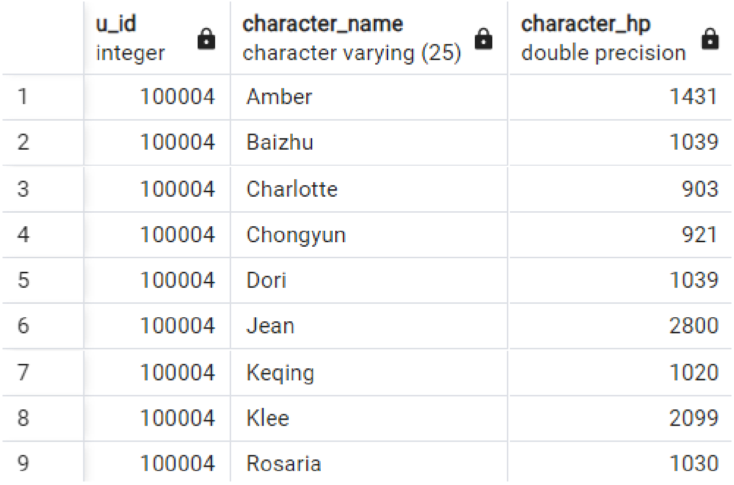
```
SELECT pc.u_id,
   pc.character_name,
   ((pc.character_level - 1) * e.hp_bonus + c.c_hp)::double precision *
    (1::double precision + COALESCE(w.w_hp, 0::double precision)) AS character_hp
  FROM player_characters pc
    JOIN characters c ON pc.character_name::text = c.character_name::text
    JOIN elements e ON c.element::text = e.element_type::text
    LEFT JOIN weapons w ON pc.character_weapon::text = w.weapon_name::text;
```

SQL:



```
SELECT *
FROM view_character_hp
where u_id = 100004
```

得到查詢結果

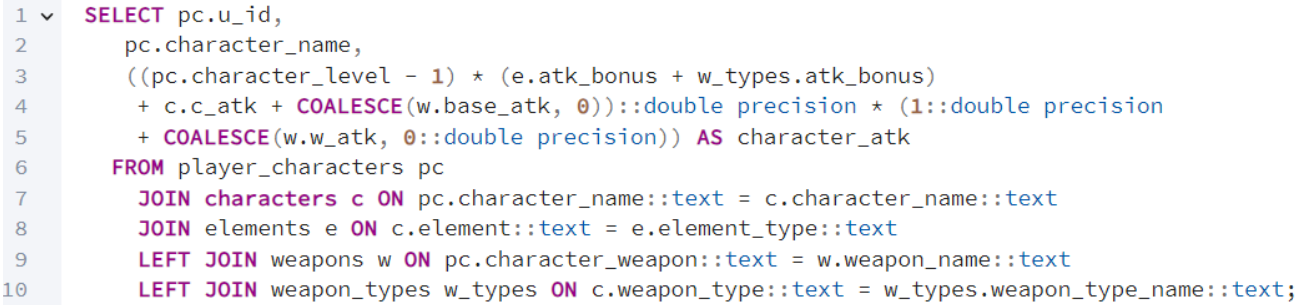


	u_id integer	character_name character varying (25)	character_hp double precision
1	100004	Amber	1431
2	100004	Baizhu	1039
3	100004	Charlotte	903
4	100004	Chongyun	921
5	100004	Dori	1039
6	100004	Jean	2800
7	100004	Keqing	1020
8	100004	Klee	2099
9	100004	Rosaria	1030

(b) view_character_atk：

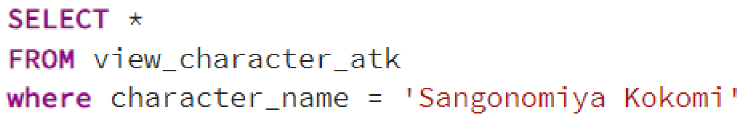
計算角色攻擊力，依照公式：

[(角色等級-1) * (角色屬性的攻擊力每等級增加數值+ 角色武器類型的攻擊力每等級增加數值)+角色等級為 1 時的基礎攻擊力] * (1+裝備武器的生命值加成%)



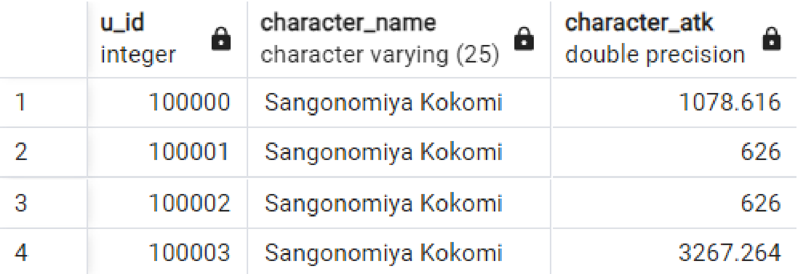
```sql
SELECT pc.u_id,
    pc.character_name,
    ((pc.character_level - 1) * (e.atk_bonus + w_types.atk_bonus)
     + c.c_atk + COALESCE(w.base_atk, 0))::double precision * (1::double precision
     + COALESCE(w.w_atk, 0::double precision)) AS character_atk
   FROM player_characters pc
     JOIN characters c ON pc.character_name::text = c.character_name::text
     JOIN elements e ON c.element::text = e.element_type::text
     LEFT JOIN weapons w ON pc.character_weapon::text = w.weapon_name::text
     LEFT JOIN weapon_types w_types ON c.weapon_type::text = w_types.weapon_type_name::text;
```

SQL:



```sql
SELECT *
FROM view_character_atk
where character_name = 'Sangonomiya Kokomi'
```

得到查詢結果

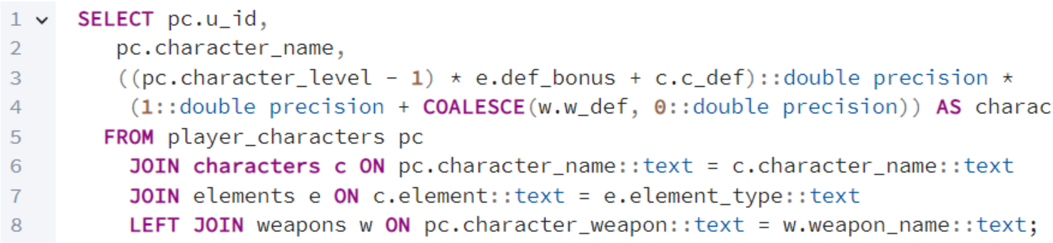


	u_id integer	character_name character varying (25)	character_atk double precision
1	100000	Sangonomiya Kokomi	1078.616
2	100001	Sangonomiya Kokomi	626
3	100002	Sangonomiya Kokomi	626
4	100003	Sangonomiya Kokomi	3267.264

(c) view_character_def：

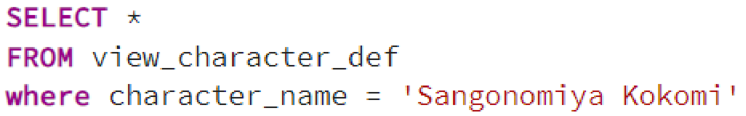
計算角色防禦力，依照公式：

[(角色等級-1) * 角色屬性的防禦力每等級增加數值+角色等級為 1 時的基礎防禦力] * (1+裝備武器的防禦力加成%)



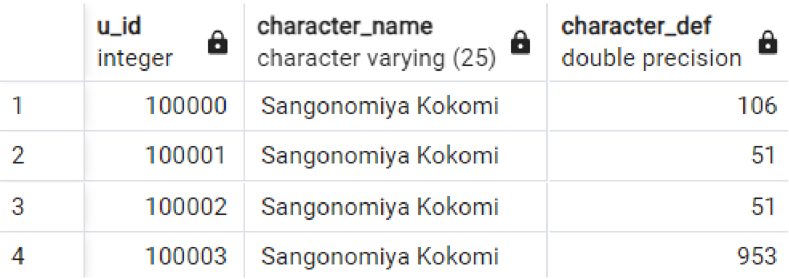
```sql
SELECT pc.u_id,
    pc.character_name,
    ((pc.character_level - 1) * e.def_bonus + c.c_def)::double precision *
     (1::double precision + COALESCE(w.w_def, 0::double precision)) AS character_def
   FROM player_characters pc
     JOIN characters c ON pc.character_name::text = c.character_name::text
     JOIN elements e ON c.element::text = e.element_type::text
     LEFT JOIN weapons w ON pc.character_weapon::text = w.weapon_name::text;
```

SQL:



```sql
SELECT *
FROM view_character_def
where character_name = 'Sangonomiya Kokomi'
```

得到查詢結果

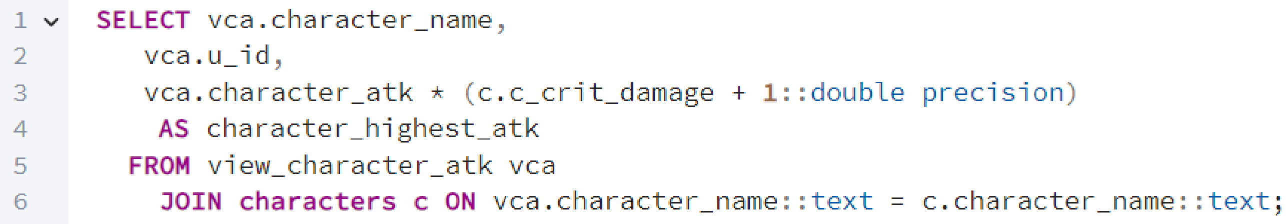


	u_id integer	character_name character varying (25)	character_def double precision
1	100000	Sangonomiya Kokomi	106
2	100001	Sangonomiya Kokomi	51
3	100002	Sangonomiya Kokomi	51
4	100003	Sangonomiya Kokomi	953

(d) view_character_highest_atk：

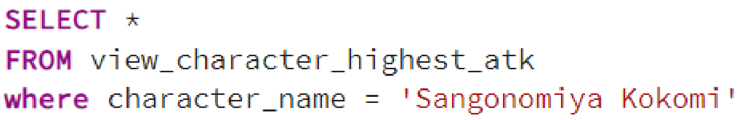
計算角色最高傷害，依照公式：

角色攻擊力*(1+爆擊傷害%)



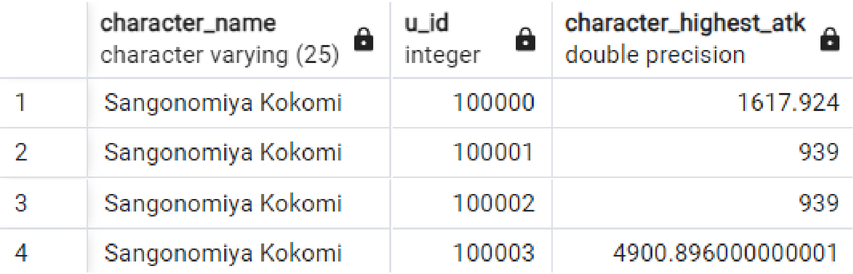
```
SELECT vca.character_name,
   vca.u_id,
   vca.character_atk * (c.c_crit_damage + 1::double precision)
    AS character_highest_atk
  FROM view_character_atk vca
    JOIN characters c ON vca.character_name::text = c.character_name::text;
```

SQL:



```
SELECT *
FROM view_character_highest_atk
where character_name = 'Sangonomiya Kokomi'
```

得到查詢結果

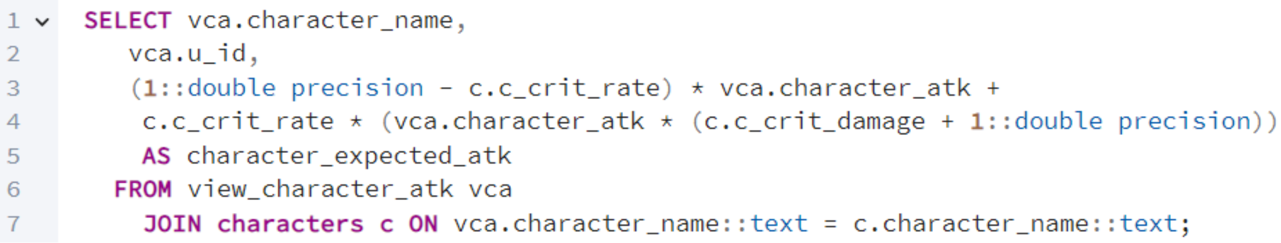


	character_name character varying (25)	u_id integer	character_highest_atk double precision
1	Sangonomiya Kokomi	100000	1617.924
2	Sangonomiya Kokomi	100001	939
3	Sangonomiya Kokomi	100002	939
4	Sangonomiya Kokomi	100003	4900.896000000001

(e) view_character_expected_atk：

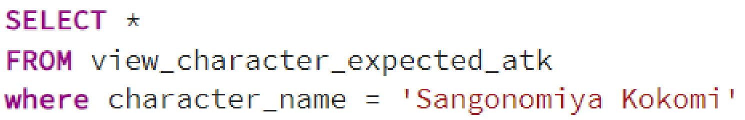
計算角色期望傷害，依照公式：

(1 - 爆擊率%)*角色最低傷害(註：為角色攻擊力)＋ 爆擊率%＊ 角色最高傷害



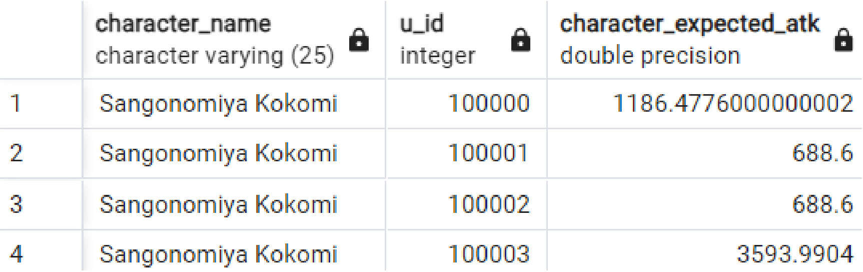
```
SELECT vca.character_name,
   vca.u_id,
   (1::double precision - c.c_crit_rate) * vca.character_atk +
    c.c_crit_rate * (vca.character_atk * (c.c_crit_damage + 1::double precision))
    AS character_expected_atk
  FROM view_character_atk vca
    JOIN characters c ON vca.character_name::text = c.character_name::text;
```

SQL:



```
SELECT *
FROM view_character_expected_atk
where character_name = 'Sangonomiya Kokomi'
```

得到查詢結果

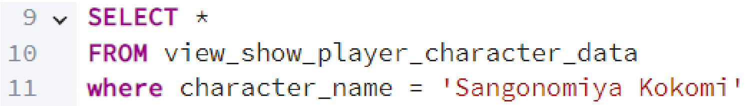


	character_name character varying (25)	u_id integer	character_expected_atk double precision
1	Sangonomiya Kokomi	100000	1186.4776000000002
2	Sangonomiya Kokomi	100001	688.6
3	Sangonomiya Kokomi	100002	688.6
4	Sangonomiya Kokomi	100003	3593.9904

(f) view_show_player_character_data：

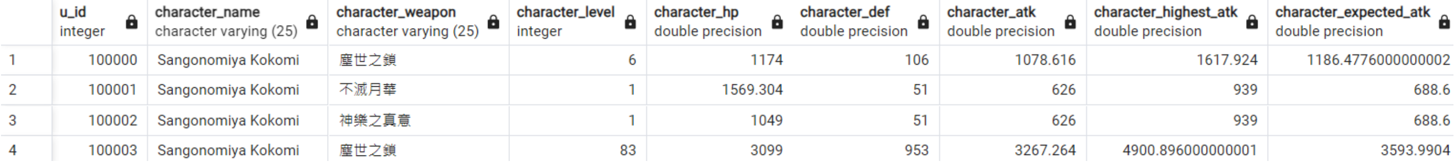
顯示完整角色資訊。

SQL：



```
SELECT *
FROM view_show_player_character_data
where character_name = 'Sangonomiya Kokomi'
```

得到查詢結果

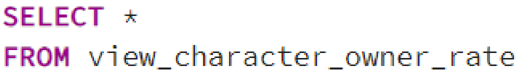


	u_id integer	character_name character varying (25)	character_weapon character varying (25)	character_level integer	character_hp double precision	character_def double precision	character_atk double precision	character_highest_atk double precision	character_expected_atk double precision
1	100000	Sangonomiya Kokomi	塵世之鎖	6	1174	106	1078.616	1617.924	1186.4776000000002
2	100001	Sangonomiya Kokomi	不滅月華	1	1569.304	51	626	939	688.6
3	100002	Sangonomiya Kokomi	神樂之真意	1	1049	51	626	939	688.6
4	100003	Sangonomiya Kokomi	塵世之鎖	83	3099	953	3267.264	4900.896000000001	3593.9904

(g) view_character_owner_rate：

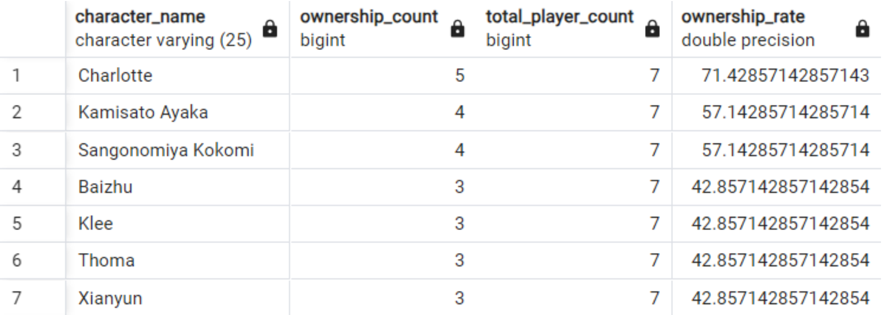
計算角色在玩家間的持有率。

SQL:



```
SELECT *
FROM view_character_owner_rate
```

得到查詢結果

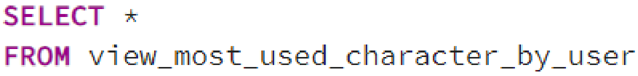


	character_name character varying (25)	ownership_count bigint	total_player_count bigint	ownership_rate double precision
1	Charlotte	5	7	71.42857142857143
2	Kamisato Ayaka	4	7	57.14285714285714
3	Sangonomiya Kokomi	4	7	57.14285714285714
4	Baizhu	3	7	42.857142857142854
5	Klee	3	7	42.857142857142854
6	Thoma	3	7	42.857142857142854
7	Xianyun	3	7	42.857142857142854

(h) view_most_used_character_by_user：

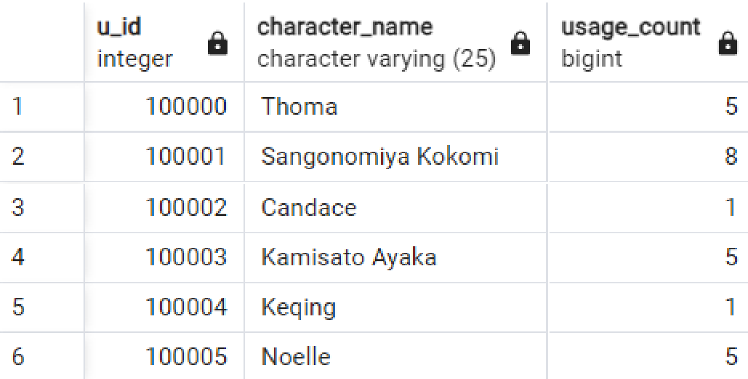
查看玩家最常使用的角色。

SQL:



```
SELECT *
FROM view_most_used_character_by_user
```

得到查詢結果

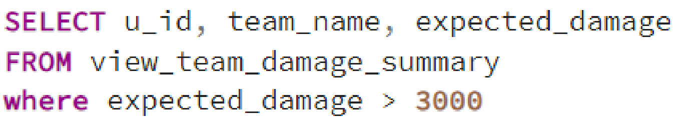


	u_id integer	character_name character varying (25)	usage_count bigint
1	100000	Thoma	5
2	100001	Sangonomiya Kokomi	8
3	100002	Candace	1
4	100003	Kamisato Ayaka	5
5	100004	Keqing	1
6	100005	Noelle	5

(i) view_team_damage_summary：

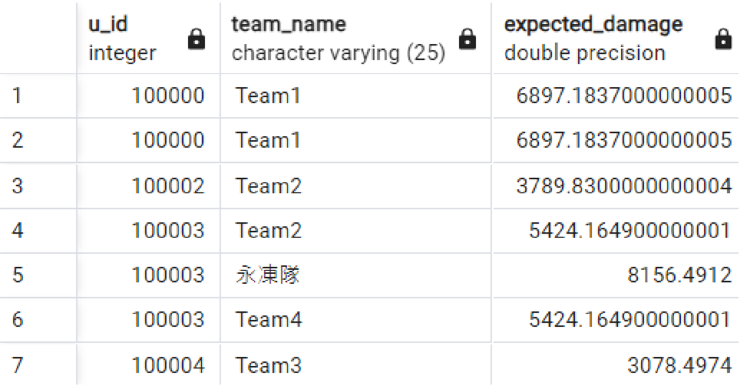
計算隊伍傷害。

SQL：



```
SELECT u_id, team_name, expected_damage
FROM view_team_damage_summary
where expected_damage > 3000
```

得到查詢結果



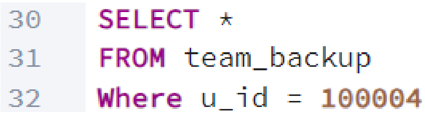
	u_id integer	team_name character varying (25)	expected_damage double precision
1	100000	Team1	6897.1837000000005
2	100000	Team1	6897.1837000000005
3	100002	Team2	3789.8300000000004
4	100003	Team2	5424.164900000001
5	100003	永凍隊	8156.4912
6	100003	Team4	5424.164900000001
7	100004	Team3	3078.4974

4.Function：

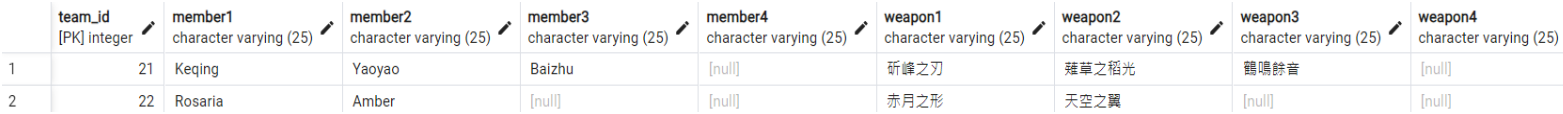
(a) create_team_from_backup(backup_team_id int, new_u_id int)：

範例一，回復以往隊伍資料：

先搜尋目前 UID = 100004 的玩家目前的備份隊伍有哪些。



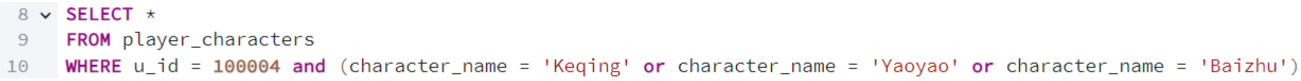
```
SELECT *
FROM team_backup
Where u_id = 100004
```



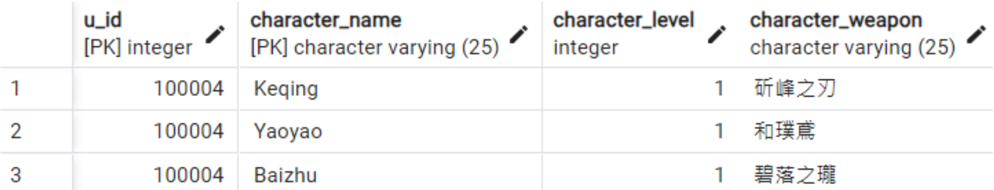
	team_id [PK] integer	member1 character varying (25)	member2 character varying (25)	member3 character varying (25)	member4 character varying (25)	weapon1 character varying (25)	weapon2 character varying (25)	weapon3 character varying (25)	weapon4 character varying (25)
1	21	Keqing	Yaoyao	Baizhu	[null]	斫峰之刃	薙草之稻光	鶴鳴餘音	[null]
2	22	Rosaria	Amber	[null]	[null]	赤月之形	天空之翼	[null]	[null]

選擇 team_id = 21，作為希望回復的隊伍。

檢查目前角色裝備設定：



```
SELECT *
FROM player_characters
WHERE u_id = 100004 and (character_name = 'Keqing' or character_name = 'Yaoyao' or character_name = 'Baizhu')
```

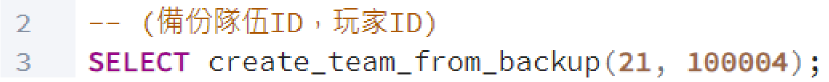


	u_id [PK] integer	character_name [PK] character varying (25)	character_level integer	character_weapon character varying (25)
1	100004	Keqing	1	斫峰之刃
2	100004	Yaoyao	1	和璞鳶
3	100004	Baizhu	1	碧落之瓏

發現，Yaoyao 跟 Baizhu 的武器跟備份上的數據不一樣。

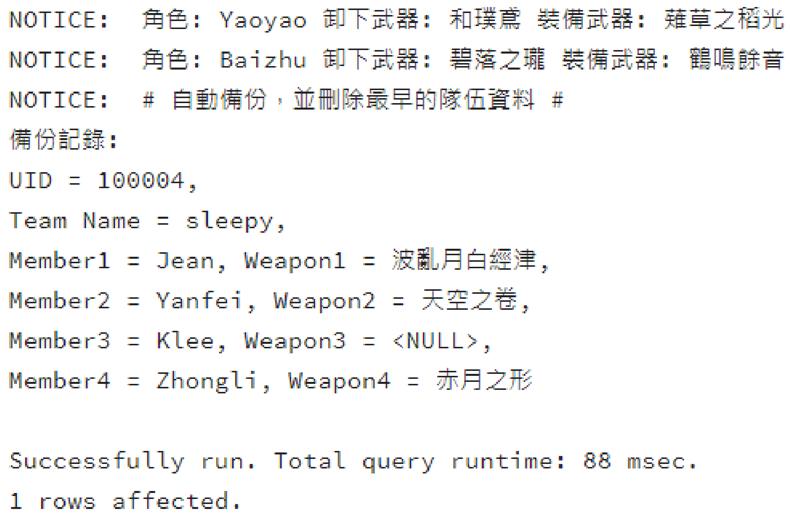
執行 function : create_team_from_backup(backup_team_id int, new_u_id int)

backup_team_id 要傳入選擇的 team_id，new_u_id 要傳入玩家的 UID



```
-- (備份隊伍ID，玩家ID)
SELECT create_team_from_backup(21, 100004);
```

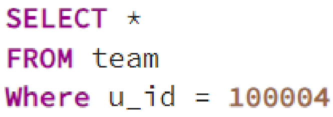
若武器不同，自動更換武器，且因為重新將備份資料中的隊伍加入目前隊伍中，因此目前隊伍中最早創建的隊伍自動備份。



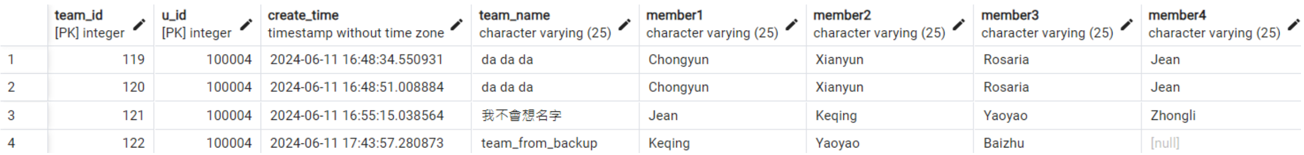
```
NOTICE:  角色: Yaoyao 卸下武器: 和璞鳶 裝備武器: 薙草之稻光
NOTICE:  角色: Baizhu 卸下武器: 碧落之瓏 裝備武器: 鶴鳴餘音
NOTICE:  # 自動備份，並刪除最早的隊伍資料 #
備份記錄:
UID = 100004,
Team Name = sleepy,
Member1 = Jean, Weapon1 = 波亂月白經津,
Member2 = Yanfei, Weapon2 = 天空之卷,
Member3 = Klee, Weapon3 = <NULL>,
Member4 = Zhongli, Weapon4 = 赤月之形

Successfully run. Total query runtime: 88 msec.
1 rows affected.
```

檢查 UID = 100004 玩家的目前隊伍資料：

多了一筆 team_from_backup 的資料。

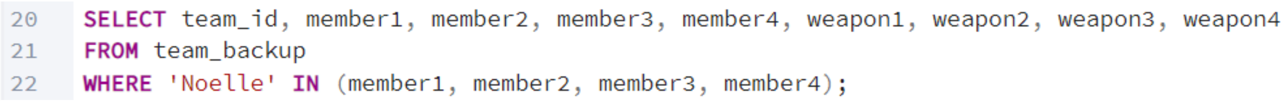


```
SELECT *
FROM team
Where u_id = 100004
```

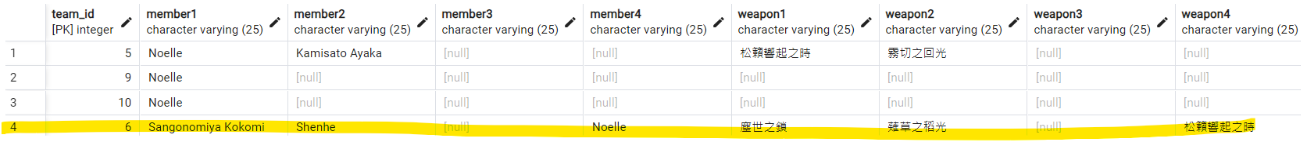


	team_id [PK] integer	u_id [PK] integer	create_time timestamp without time zone	team_name character varying (25)	member1 character varying (25)	member2 character varying (25)	member3 character varying (25)	member4 character varying (25)
1	119	100004	2024-06-11 16:48:34.550931	da da da	Chongyun	Xianyun	Rosaria	Jean
2	120	100004	2024-06-11 16:48:51.008884	da da da	Chongyun	Xianyun	Rosaria	Jean
3	121	100004	2024-06-11 16:55:15.038564	我不會想名字	Jean	Keqing	Yaoyao	Zhongli
4	122	100004	2024-06-11 17:43:57.280873	team_from_backup	Keqing	Yaoyao	Baizhu	[null]

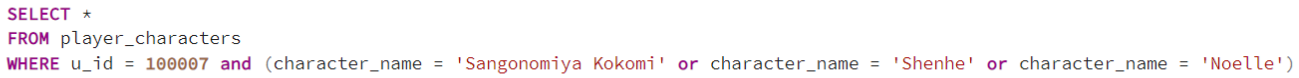
範例二，使用他人曾創建的隊伍資料創建自己的隊伍：



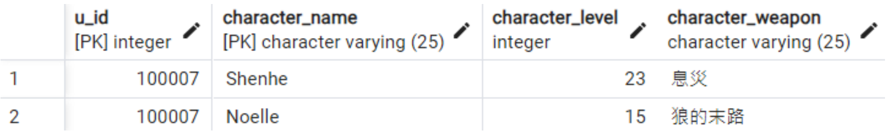
```sql
SELECT team_id, member1, member2, member3, member4, weapon1, weapon2, weapon3, weapon4
FROM team_backup
WHERE 'Noelle' IN (member1, member2, member3, member4);
```



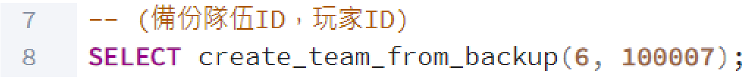
	team_id [PK] integer	member1 character varying (25)	member2 character varying (25)	member3 character varying (25)	member4 character varying (25)	weapon1 character varying (25)	weapon2 character varying (25)	weapon3 character varying (25)	weapon4 character varying (25)
1	5	Noelle	Kamisato Ayaka	[null]	[null]	松籟響起之時	霧切之回光	[null]	[null]
2	9	Noelle	[null]	[null]	[null]	[null]	[null]	[null]	[null]
3	10	Noelle	[null]	[null]	[null]	[null]	[null]	[null]	[null]
4	6	Sangonomiya Kokomi	Shenhe	[null]	Noelle	塵世之鎖	薙草之稻光	[null]	松籟響起之時



```sql
SELECT *
FROM player_characters
WHERE u_id = 100007 and (character_name = 'Sangonomiya Kokomi' or character_name = 'Shenhe' or character_name = 'Noelle')
```

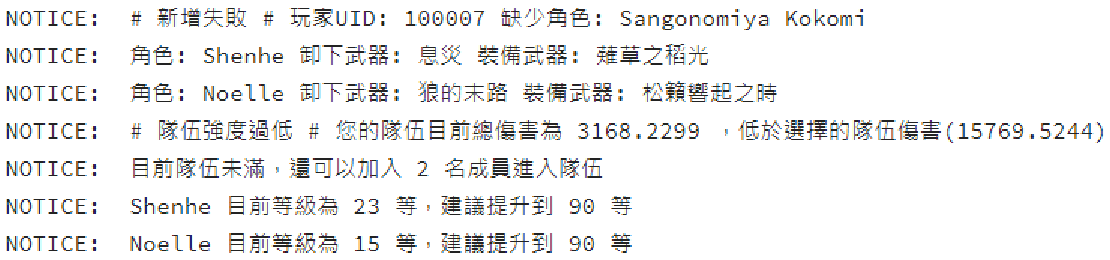


	u_id [PK] integer	character_name [PK] character varying (25)	character_level integer	character_weapon character varying (25)
1	100007	Shenhe	23	息災
2	100007	Noelle	15	狼的末路



```sql
-- (備份隊伍ID，玩家ID)
SELECT create_team_from_backup(6, 100007);
```

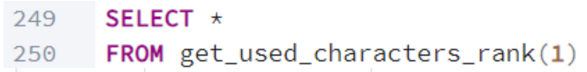
若希望創建的隊伍中有自己沒有的角色，則在創建過程中自動掠過，並輸出提示”缺少角色”，且會比較強度差距，若低於預期，則輸出建議。



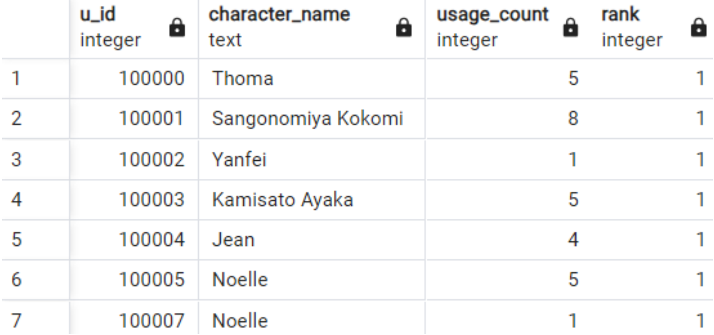
```
NOTICE:  # 新增失敗 # 玩家UID: 100007 缺少角色: Sangonomiya Kokomi
NOTICE:  角色: Shenhe 卸下武器: 息災 裝備武器: 薙草之稻光
NOTICE:  角色: Noelle 卸下武器: 狼的末路 裝備武器: 松籟響起之時
NOTICE:  # 隊伍強度過低 # 您的隊伍目前總傷害為 3168.2299 ，低於選擇的隊伍傷害(15769.5244)
NOTICE:  目前隊伍未滿，還可以加入 2 名成員進入隊伍
NOTICE:  Shenhe 目前等級為 23 等，建議提升到 90 等
NOTICE:  Noelle 目前等級為 15 等，建議提升到 90 等
```

(b) get_used_characters_rank(rank_n int)：

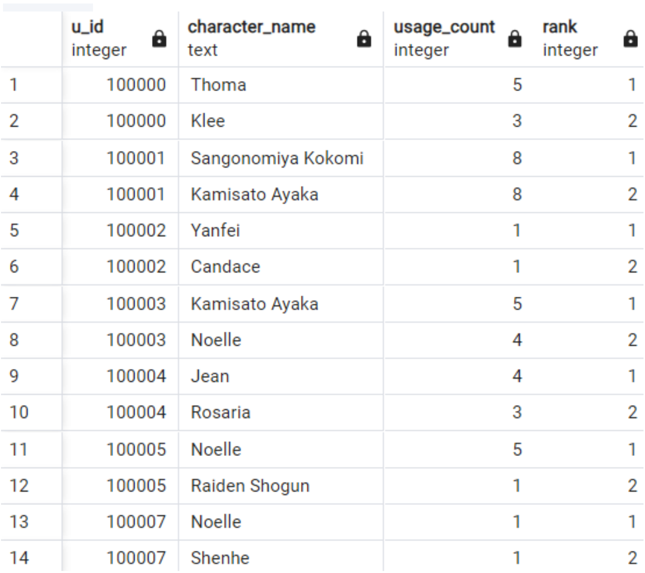
輸出每個玩家前 rank_n 個最常使用的角色，若使用次數一樣則，名字排序的先後順序作為排名依據。



```sql
SELECT *
FROM get_used_characters_rank(1)
```



	u_id integer	character_name text	usage_count integer	rank integer
1	100000	Thoma	5	1
2	100001	Sangonomiya Kokomi	8	1
3	100002	Yanfei	1	1
4	100003	Kamisato Ayaka	5	1
5	100004	Jean	4	1
6	100005	Noelle	5	1
7	100007	Noelle	1	1



	u_id integer	character_name text	usage_count integer	rank integer
1	100000	Thoma	5	1
2	100000	Klee	3	2
3	100001	Sangonomiya Kokomi	8	1
4	100001	Kamisato Ayaka	8	2
5	100002	Yanfei	1	1
6	100002	Candace	1	2
7	100003	Kamisato Ayaka	5	1
8	100003	Noelle	4	2
9	100004	Jean	4	1
10	100004	Rosaria	3	2
11	100005	Noelle	5	1
12	100005	Raiden Shogun	1	2
13	100007	Noelle	1	1
14	100007	Shenhe	1	2