《数据库概论》实验一：用SQL进行数据操作 实验报告

姓名：郑凯琳
学号：205220025
联系方式：205220025@smail.nju.edu.cn

一、实验环境

操作系统：Windows 11
mysql-workbench-community-8.0.38-winx64

二、实验过程

Q1

```
SELECT COUNT(*) AS 'speciesCount'
FROM species
WHERE description LIKE '%this%';
```

> **思路：** 使用 `LIKE` 语句进行字符串匹配， `%` 表示匹配任意长度的字符串。

查询结果：

	speciesCount
▶	90

Q2

```
SELECT username, SUM(PH.power)AS totalPhonemonPower
FROM player P
JOIN phonemon PH ON P.id = PH.player
WHERE P.username IN ('Cook', 'Hughes')
GROUP BY P.username;
```

> **思路：** 计算玩家 'Cook' 和 'Hughes' 各自拥有的 Phonemon 总能量。
>
> - 使用 `JOIN` 连接 `Player` 表和 `Phonemon` 表。
> - `WHERE` 子句筛选用户名为 'Cook' 和 'Hughes' 的记录。

- 使用 `SUM(Phonemon.power)` 计算能量总和，并按用户名分组。

查询结果：

	username	totalPhonemonPower
▶	Cook	1220
	Hughes	1170

Q3

```
SELECT T.title, COUNT(P.id) AS numberOfPlayers
FROM team T
JOIN player P ON T.id = P.team
GROUP BY T.title
ORDER BY numberOfPlayers DESC;
```

> **思路：** 统计每个队伍的成员数量。
>
> - 使用 `JOIN` 连接 `Player` 表和 `Team` 表。
> - `COUNT(Player.id)` 计算每个队伍的玩家数量。
> - `GROUP BY Team.id` 按队伍 `id` 分组，结果按玩家数量降序排列。

查询结果：

	title	numberOfPlayers
▶	Mystic	8
	Valor	6
	Instinct	5

Q4

```
SELECT S.id AS idSpecies, S.title
FROM species S
JOIN type T ON S.type1 = T.id OR S.type2 = T.id
WHERE T.title = 'grass';
```

> **思路：** 查询所有类型为 'grass' 的物种。
>
> - 使用 `JOIN` 连接 `Species` 表和 `Type` 表。
> - 筛选条件 `WHERE Type.title = 'grass'` 用于获取相关物种。
> - 考虑物种的两个类型外键，使用 `OR` 进行条件匹配。

查询结果：

	idSpecies	title
▶	1	Bulbasaur
	2	Ivysaur
	3	Venusaur
	43	Oddish
	44	Gloom
	45	Vileplume
	69	Bellsprout
	70	Weepinbell
	71	Victreebel
	102	Exeggcute
	103	Exeggutor
	114	Tangela

Q5

```sql
SELECT P.id AS idPlayer, P.username
FROM player P
WHERE P.id NOT IN(
    SELECT PU.player
    FROM purchase PU
    JOIN item I ON PU.item = I.id
    WHERE I.type = 'F'
);
```

> **思路：** 找出从未购买过食物的玩家。
>
> - 使用子查询（`Player`，`Item`，`Purchase` 的联结表）筛选出已购买食物的玩家 `id`。
> - `NOT IN` 谓词用于从 `Player` 表中选择未购买食物的玩家。

查询结果：

	idPlayer	username
▶	4	Reid
	7	Hughes
	8	Bruce
	10	Lyons
	11	Emily
	12	Darthy
	15	Huma

Q6

```sql
SELECT P.level, SUM(PU.quantity * I.price) AS totalAmountSpentByAllPlayersAtLevel
FROM player P
JOIN purchase PU ON P.id = PU.player
JOIN item I ON PU.item = I.id
GROUP BY P.level
ORDER BY totalAmountSpentByAllPlayersAtLevel DESC;
```

> **思路：** 按等级统计玩家的总消费金额。
>
> - 连接 `Player` 、 `Purchase` 和 `Item` 表。
> - 使用 `SUM(Purchase.quantity * Item.price)` 计算每个等级的总消费。
> - 按等级分组，并按总消费金额降序排列。

查询结果：

	level	totalAmountSpentByAllPlayersAtLevel
▶	2	130.68
	12	95.45
	6	62.37
	5	52.98
	3	51.75
	1	39.58
	4	33.74
	8	29.48
	11	26.97
	7	24.26
	10	17.22
	9	9.99

Q7

```
SELECT I.id AS item, i.title, COUNT(PU.id) AS numTimesPurchased
FROM purchase PU
JOIN item I ON PU.item = I.id
GROUP BY I.id
HAVING COUNT(PU.id) = (
    SELECT MAX(counts.num)
    FROM (SELECT COUNT(*) AS num FROM purchase GROUP BY item) counts
);
```

> **思路：** 找出购买次数最多的物品。
>
> - 使用 JOIN 连接 Purchase 表和 Team 表。
> - COUNT(purchase.item) 统计每个物品的购买次数。
> - 使用 HAVING 子句筛选出购买次数大于等于最大次数的物品。

查询结果：

	item	title	numTimesPurchased
▶	1	Phoneball	10

Q8

```
SELECT P.id AS playerID, P.username, COUNT(DISTINCT I.id) AS numberDistinctFoodItemsPurchased
FROM player P
JOIN purchase PU ON P.id = PU.player
JOIN item I ON PU.item = I.id
WHERE I.type = 'F'
GROUP BY P.id, P.username
HAVING COUNT(DISTINCT I.id) = (
    SELECT COUNT(DISTINCT id) FROM item WHERE type = 'F'
);
```

> **思路：** 找出购买所有食品的玩家。
>
> - 使用 JOIN 连接 Player 、 Purchase 和 Item 表。
> - 统计每个玩家购买的独特食品种类数量。
> - 使用 HAVING 确保该数量等于所有食品种类的数量。降序排列。

查询结果：

	playerID	username	numberDistinctFoodItemsPurchased
▶	20	Zihan	6

Q9

```sql
SELECT
    COUNT(*) AS numberOfPhonemonPairs,
    ROUND(SQRT(POWER((P1.latitude - P2.latitude), 2) + POWER((P1.longitude - P2.longitude), 2))
FROM phonemon P1, phonemon P2
WHERE P1.id < P2.id
GROUP BY distanceX
HAVING distanceX <= ALL(
    SELECT
        ROUND(SQRT(POWER((P1.latitude - P2.latitude), 2) + POWER((P1.longitude - P2.longitude),
    FROM phonemon P1, phonemon P2
    WHERE P1.id < P2.id
);
```

> **思路：** 计算距离最近的 Phonemon 对的数量。
>
> - 自联结 Phonemon 表计算所有 Phonemon 对的欧氏距离。
> - 使用 `WHERE t1.id < t2.id` 确保不重复计算。
> - 按距离分组，使用 HAVING 筛选出具有最小距离的对。

查询结果：

	numberOfPhonemonPairs	distanceX
▶	98	0.19

Q10

```
SELECT T.username, T.typeTitle
FROM
(
        SELECT player.username AS username, type.title AS typeTitle, type.id
        FROM player, phonemon, species, type
        WHERE player.id = phonemon.player
            AND phonemon.species = species.id
            AND (species.type1 = type.id OR species.type2 = type.id)
        GROUP BY player.id, type.id
        HAVING COUNT(DISTINCT species.id) = (
            SELECT COUNT(*) FROM species WHERE species.type1 = type.id or species.type2 = type.
        )
)AS T
```

> **思路：** 找出捕捉到特定类型物种的玩家。
>
> - 连接 `Player` 、 `Phonemon` 、 `Species` 和 `Type` 表。
> - 按玩家和类型分组，统计每个玩家捕捉到的物种数量。
> - 使用 `HAVING` 确保数量与该类型所有物种的数量一致。

查询结果：

	username	typeTitle
▶	Lyons	Bug
	Lyons	Fairy

三、实验中遇到的困难及解决办法

困难：

编写形式简单、易于理解的SQL语句

解决办法：

尽可能利用单次循环可得的信息；合理高效地使用统计方法