《数据库概论》实验一：用 SQL 进行数据操作 实验报告

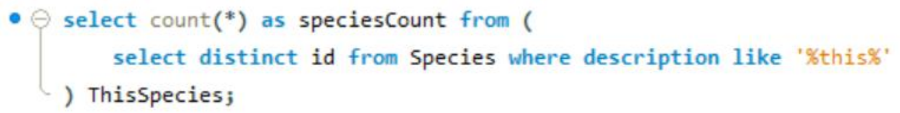
张涵之　191220154　191220154@smail.nju.edu.cn

实验环境

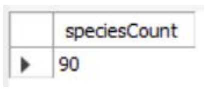
操作系统：Windows 10，64 位操作系统，基于 x64 的处理器
软件版本：MySQL Community 8.0.26.0

实验过程

1. 有多少 species 的 description 中含有单词“this”？返回：(speciesCount)
   直接使用通配符与 species 中的 description 进行匹配，用 count()统计

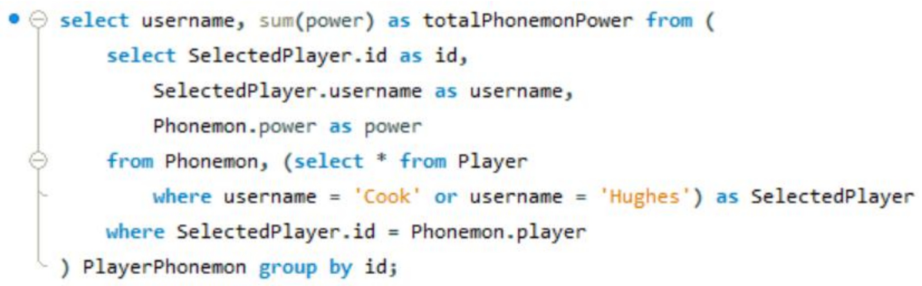


```
select count(*) as speciesCount from (
    select distinct id from Species where description like '%this%'
) ThisSpecies;
```

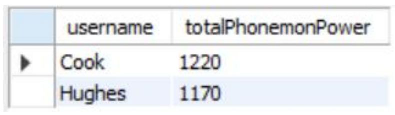


	speciesCount
▶	90

2. 对于 player ‘Cook’ 与 player ‘Hughes’，显示他们的 username 和各自拥有的 Phonemon 的总能量。返回：(username, totalPhonemonPower)
   先选出 username 为 ‘Cook’ 和 ‘Hughes’ 的 player 为 SelectedPlayer
   再从 SelectedPlayer 和 Phonemon 中选出两个被两个 player 捕捉的 phonemon
   将这些 phonemon 按 player id 分组，用 sum()计算 power 的和

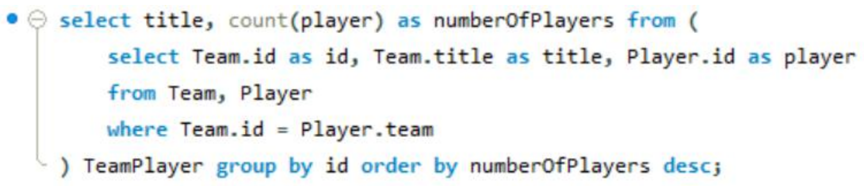


```
select username, sum(power) as totalPhonemonPower from (
    select SelectedPlayer.id as id,
        SelectedPlayer.username as username,
        Phonemon.power as power
    from Phonemon, (select * from Player
        where username = 'Cook' or username = 'Hughes') as SelectedPlayer
    where SelectedPlayer.id = Phonemon.player
) PlayerPhonemon group by id;
```

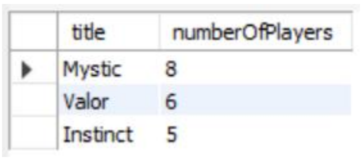


	username	totalPhonemonPower
▶	Cook	1220
	Hughes	1170

3. 每个 team 有多少名 player？按玩家数量降序返回：(title, numberOfPlayers)
   先按 team id 和 player team 连接，对结果进行分组 count()，降序输出

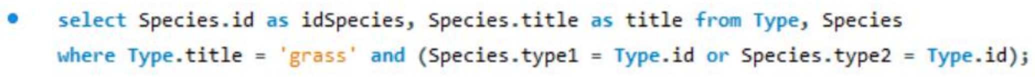


```
select title, count(player) as numberOfPlayers from (
    select Team.id as id, Team.title as title, Player.id as player
    from Team, Player
    where Team.id = Player.team
) TeamPlayer group by id order by numberOfPlayers desc;
```

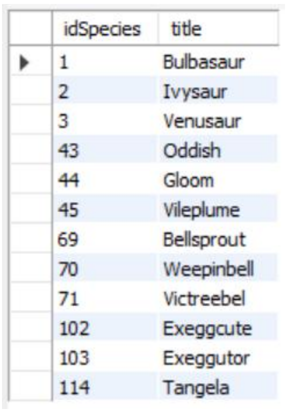


	title	numberOfPlayers
▶	Mystic	8
	Valor	6
	Instinct	5

4. 哪些 species 具有 type‘grass’？返回：(idSpecies, title)
从 Species 和 Type 中找到 species type1 对应 type id 的 type title 为‘grass’或 type2 对应的 type id 的 type title 为‘grass’的并输出

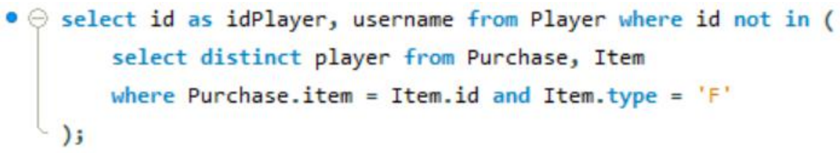


```
• select Species.id as idSpecies, Species.title as title from Type, Species
  where Type.title = 'grass' and (Species.type1 = Type.id or Species.type2 = Type.id);
```

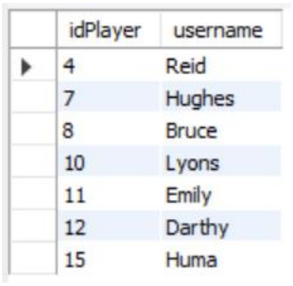


	idSpecies	title
▶	1	Bulbasaur
	2	Ivysaur
	3	Venusaur
	43	Oddish
	44	Gloom
	45	Vileplume
	69	Bellsprout
	70	Weepinbell
	71	Victreebel
	102	Exeggcute
	103	Exeggutor
	114	Tangela

5. 列出从未购买过 food 的 player。返回：(idPlayer, username)
先找出所有在 Purchase 中有对应 item 的 type 为‘F’的 player 的 id
即所有购买过 food 的 player，再找出不在这个集合中的 player

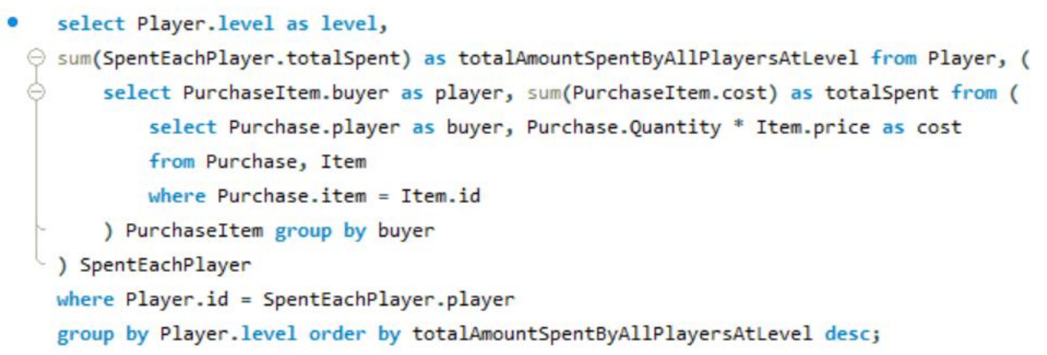


```
• select id as idPlayer, username from Player where id not in (
      select distinct player from Purchase, Item
      where Purchase.item = Item.id and Item.type = 'F'
  );
```

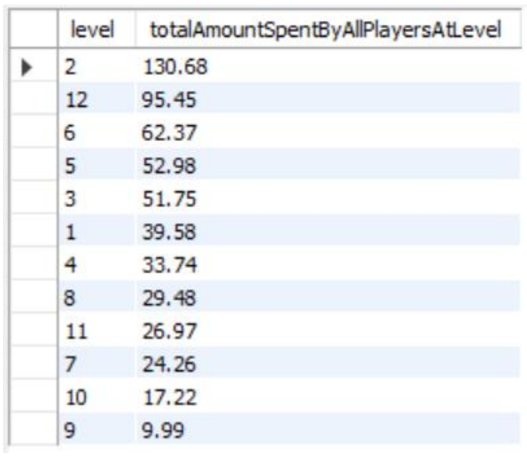


	idPlayer	username
▶	4	Reid
	7	Hughes
	8	Bruce
	10	Lyons
	11	Emily
	12	Darthy
	15	Huma

6. 以金额降序列出每一特定 level 以及该等级的所有 player 在购买上花费的总金额。返回：(level, totalAmountSpentByAllPlayersAtLevel)
先将 Purchase 与 Item 对应，按 quantity 和 price 算出每个 purchase 的金额
再按 purchase 的 player id 进行分组，算出每个 player 消费总金额
然后对 SpentEachPlayer 按 player 的 level 分组，算出每个 level 消费总金额
最后按每个 level 对应的 totalAmountSpentByAllPlayersAtLevel 降序输出

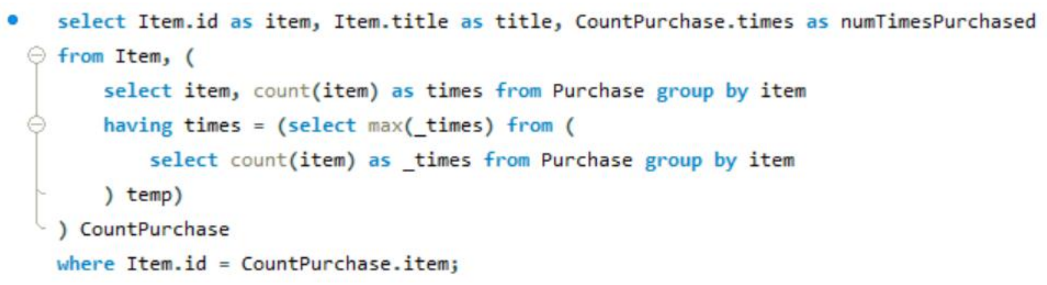


```
select Player.level as level,
sum(SpentEachPlayer.totalSpent) as totalAmountSpentByAllPlayersAtLevel from Player, (
    select PurchaseItem.buyer as player, sum(PurchaseItem.cost) as totalSpent from (
        select Purchase.player as buyer, Purchase.Quantity * Item.price as cost
        from Purchase, Item
        where Purchase.item = Item.id
    ) PurchaseItem group by buyer
) SpentEachPlayer
where Player.id = SpentEachPlayer.player
group by Player.level order by totalAmountSpentByAllPlayersAtLevel desc;
```

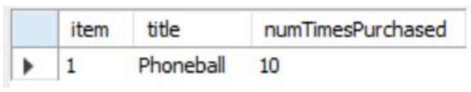


level	totalAmountSpentByAllPlayersAtLevel
2	130.68
12	95.45
6	62.37
5	52.98
3	51.75
1	39.58
4	33.74
8	29.48
11	26.97
7	24.26
10	17.22
9	9.99

7. 什么 item 购买次数最多（含并列）？返回：(item, title, numTimesPurchased)
先按 purchase 的 item id 分组找出购买次数最多的 item 被购买的次数
再找出所有购买次数等于这个值的 item，输出 id，title 和购买次数

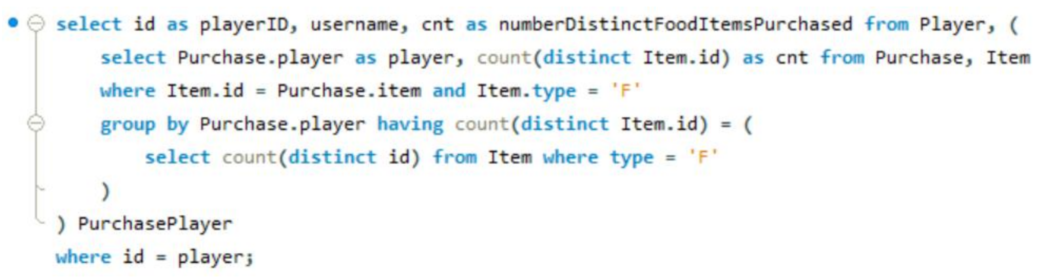


```
select Item.id as item, Item.title as title, CountPurchase.times as numTimesPurchased
from Item, (
    select item, count(item) as times from Purchase group by item
    having times = (select max(_times) from (
        select count(item) as _times from Purchase group by item
    ) temp)
) CountPurchase
where Item.id = CountPurchase.item;
```

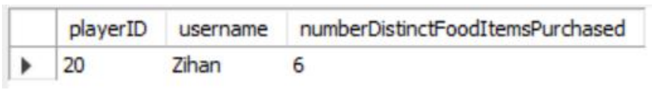


item	title	numTimesPurchased
1	Phoneball	10

8. 找到可获取的食物的数量，和购买所有种类食物至少各一次的玩家。返回：(playerID, username, numberDistinctFoodItemsPurchased)
首先用 count()统计 Item 中 type 为‘F’的数量，即“可获取的食物数量”（理论上，Item 中 type 为‘F’的 id 应该与 Food 中的 id 一一对应，否则如果某个 Food 的 id 在 item 中找不到对应，则它是一个不可（通过购买）获取的食物，因为 Purchase 中的 id 是标记 item 的。故这里不直接统计 Food）
对 Purchase 取 item type 为‘F’的，按 player 进行分组，
用 count distinct 根据 purchase 中的 item id 统计每个 player 买过的食物种数
取购买食物种数等于“可获取食物数量”的 player，输出 id 和 username

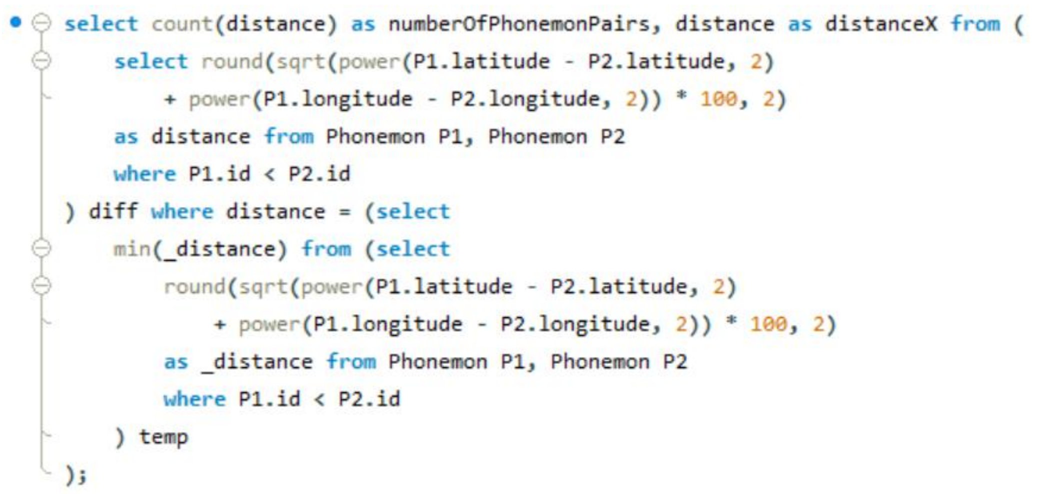


```
select id as playerID, username, cnt as numberDistinctFoodItemsPurchased from Player, (
    select Purchase.player as player, count(distinct Item.id) as cnt from Purchase, Item
    where Item.id = Purchase.item and Item.type = 'F'
    group by Purchase.player having count(distinct Item.id) = (
        select count(distinct id) from Item where type = 'F'
    )
) PurchasePlayer
where id = player;
```

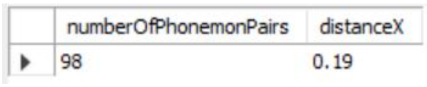


	playerID	username	numberDistinctFoodItemsPurchased
▶	20	Zihan	6

9. 将距离最近的两个 Phonemon 之间的欧氏距离称为 X。计算相互之间距离为 X 的 Phonemon 对的数量。返回：(numberOfPhonemonPairs,distanceX)
对 Phonemon 取副本 P1、P2，通过 P1.id < P2.id 防止 pair 的重复计算
对每个 pair 用公式算出最近欧氏距离 X，用 round()保留两位小数
再计算一次每对之间距离，取等于 X 的对并进行计数

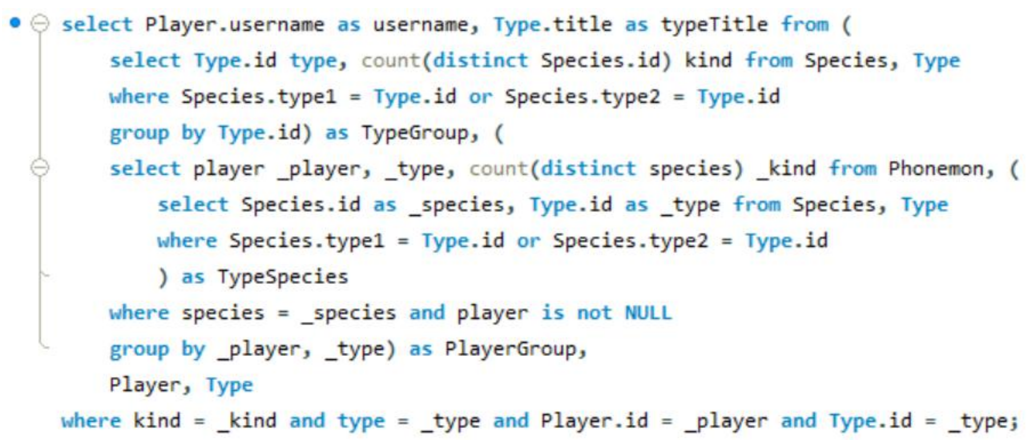


```
select count(distance) as numberOfPhonemonPairs, distance as distanceX from (
    select round(sqrt(power(P1.latitude - P2.latitude, 2)
        + power(P1.longitude - P2.longitude, 2)) * 100, 2)
    as distance from Phonemon P1, Phonemon P2
    where P1.id < P2.id
) diff where distance = (select
    min(_distance) from (select
        round(sqrt(power(P1.latitude - P2.latitude, 2)
        + power(P1.longitude - P2.longitude, 2)) * 100, 2)
        as _distance from Phonemon P1, Phonemon P2
        where P1.id < P2.id
    ) temp
);
```

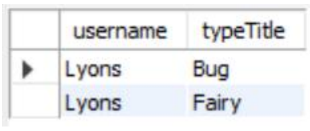


	numberOfPhonemonPairs	distanceX
▶	98	0.19

10. 列出捕捉了任一特定 type 中每一 species 至少各一个 Phonemon 的玩家的名称以及该类型的名称。返回：(username, typeTitle)
先根据 Type，Species 找出每种 type 拥有的 species 种数
从 Player，Type，Phonemon 找出每个 player 拥有每个 type 的 species 种数
从 TypeGroup，PlayerGroup 选出 type 和对应 species 种数相同的组
结合 Type 和 Player 输出这些组的 player username 和 type title



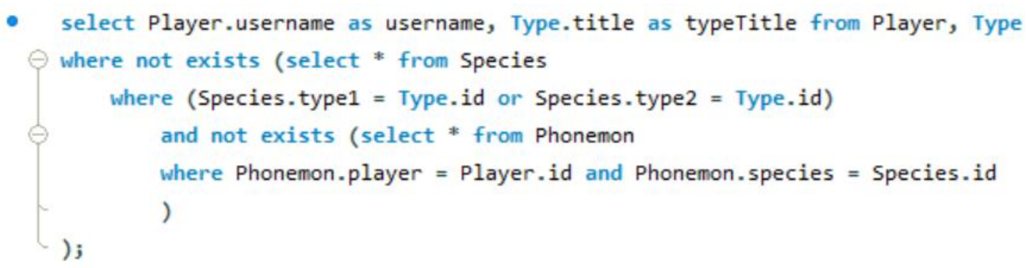
```
select Player.username as username, Type.title as typeTitle from (
    select Type.id type, count(distinct Species.id) kind from Species, Type
    where Species.type1 = Type.id or Species.type2 = Type.id
    group by Type.id) as TypeGroup, (
    select player _player, _type, count(distinct species) _kind from Phonemon, (
        select Species.id as _species, Type.id as _type from Species, Type
        where Species.type1 = Type.id or Species.type2 = Type.id
        ) as TypeSpecies
    where species = _species and player is not NULL
    group by _player, _type) as PlayerGroup,
    Player, Type
where kind = _kind and type = _type and Player.id = _player and Type.id = _type;
```



	username	typeTitle
▶	Lyons	Bug
	Lyons	Fairy

实验中遇到的困难及解决办法

1. 有些表定义的域在手册中找不到解释，如 Food 的 manna 不知道是什么有什么用，令人有些困惑，虽然对实验本身没有什么影响；
2. 实验任务部分比较难读，需要结合上下文实际代码进行理解，如“可获取的食物的数量”究竟该统计 Food 还是 Item 中的‘F’取决于具体含义；
3. 数据库规模较大，输出结果是否正确很难手动验算，可以考虑根据不同的逻辑实现同一个任务，并将输出结果进行比较，如 10 中找出集齐了特定 type 中所有 species 的 player，上面已经用计算每个 type 下 species 数量和每个玩家捕捉过每个 type 的 species 数量进行比对的方法实现。除此之外，还可以单独对每个 player，type 组合判断是否存在某一个被这个 type 包含而 player 没有捕捉过的 species（即没有对应 player 的 phonemon 属于这个 species），如果不存在这样的 species，那么这个 player 就集齐了这个 type，否则没有集齐。这种方法的输出结果与上面的相同，其他一些较复杂的任务也可以用这种方法进行验证，从而最大程度上保证代码运行结果的正确性。



```
select Player.username as username, Type.title as typeTitle from Player, Type
where not exists (select * from Species
    where (Species.type1 = Type.id or Species.type2 = Type.id)
        and not exists (select * from Phonemon
        where Phonemon.player = Player.id and Phonemon.species = Species.id
        )
);
```

参考文献及致谢

课程课件（主要是 database_03_2.pdf）
部分函数使用（如 round()的参数和具体用法）参考了网络，如 runoob.com 的 SQL 教程部分：https://www.runoob.com/sql/sql-tutorial.html