

# Групиране и агрегация

Упражнение 5

# Задача

Да се преброят корабите във всеки клас (да извлечеме таблица с 2 колони - име на клас, брой кораби от този клас)

CLASS	NAME
Iowa	Fuso
Iowa	Hood
Iowa	Iowa
Iowa	King George V
Iowa	New Jersey
Iowa	Missouri
Iowa	West Virginia
Iowa	Wisconsin
Kongo	Kirishima
Kongo	Kongo
Kongo	Haruna
Kongo	Hiei
North Carolina	Bismarck
North Carolina	North Carolina
North Carolina	Prince of Wales
North Carolina	Schamhorst
North Carolina	South Dakota
North Carolina	Washington
Renown	Duke of York
Renown	Renown
Renown	Repulse
Renown	Resolution
Revenge	Revenge
Revenge	Ramillies
Revenge	Royal Oak
Revenge	Royal Sovereign
Tennessee	Tennessee
Tennessee	California
Yamato	Rodney
Yamato	Musashi
Yamato	Yamashiro
Yamato	Yamato
Yamato	Yanashiro

Iowa, 8

Kongo, 4

North Carolina, 6

Renown, 4

Revenge, 4

Tennessee, 2

Yamato, 5

```
SELECT c.CLASS, s.NAME  
FROM CLASSES c  
      JOIN SHIPS s ON c.CLASS = s.CLASS  
ORDER BY c.CLASS
```

```
SELECT c.CLASS, COUNT(s.NAME) AS NUMSHIPS  
FROM CLASSES c  
      JOIN SHIPS s ON c.CLASS = s.CLASS  
GROUP BY c.CLASS
```

# Агрегатни функции

Извършват пресмятане върху множество от стойности (например множеството от стойности в някоя колона за група формирана чрез GROUP BY) и връщат скалар.

- ❑ COUNT ([ ALL | DISTINCT ] expression)
- ❑ SUM ([ ALL | DISTINCT ] expression)
- ❑ AVG ([ ALL | DISTINCT ] expression)
- ❑ MIN (expression)
- ❑ MAX (expression)
- ❑ STRING\_AGG (expression, delimiter)  
(GROUP\_CONCAT в MySQL; LISTAGG в Oracle и DB2)

Агрегатните функции игнорират NULL стойностите.

ALL е по подразбиране. С DISTINCT може да игнорираме повтарящи се стойности).

Специален случай - COUNT(\*) - NULL стойностите ще бъдат преброени, както и повторенията.

# GROUP BY клауза

- ❑ В примера по-горе, видяхме групиране по стойности в колоната NAME от таблицата SHIPS. В общия случай, групирането може да става по един или повече изрази (в частност по една или повече колони).
- ❑ Два реда от резултата, формиран след изпълнението на FROM и WHERE клаузите, попадат в една и съща група, ако стойностите на съответните изрази в GROUP BY клаузите съвпадат за тях.
- ❑ NULL стойности могат да формират група (при групиране, не се игнорират)
- ❑ Примери:
  1. Групиране по 2 колони: `GROUP BY s.CLASS, s.LAUNCHED` - в една и съща група ще попаднат кораби от един и същи клас, които са пуснати на вода в една и съща година.
  2. Групиране по израз: `GROUP BY YEAR(BATTLES.DATE)` - в една и съща група ще попаднат кораби, които са участвали в битка в една и съща година

# HAVING клауза

Аналогична на WHERE клаузата. Дава възможност да филтрираме, след като сме извършили групиране.

Пример: Да се извлече броя корабите във всеки клас, който има повече от 2 кораба

```
SELECT c.CLASS, COUNT(s.NAME) AS NUMSHIPS
FROM CLASSES c
      JOIN SHIPS s ON c.CLASS = s.CLASS
GROUP BY c.CLASS
HAVING COUNT(s.NAME) > 2
```

# Логически ред на изпълнение на SQL заявка

1. Извършват се свързванията, описани във FROM клаузата от ляво надясно
2. Извършва се филтрирането на редове формирани след свързванията в WHERE клаузата
3. Извършва се групиране на останалите редове в групи на базата на GROUP BY клаузата. Всяка група формира по един ред в крайния резултат.
4. Извършва се филтриране на редове след извършване на групирането в HAVING клаузата
5. Формират се колоните, описани във SELECT клаузата, които ще бъдат върнати на клиента
6. Извършва се сортирането описано в ORDER BY клаузата
7. Резултатът се връща на клиента, изпратил заявката

**ВАЖНО:** След групиране, в SELECT, HAVING и ORDER BY може да използваме само колоните или изразите, по които сме групирали или изрази формирани от такива колони и/или агрегатни функции.

# Оператор CASE

```
CASE input_expression  
  WHEN when_expression THEN result_expression [ ...n ]  
  [ ELSE else_result_expression ]  
END
```

```
CASE  
  WHEN boolean_expression THEN result_expression [ ...n ]  
  [ ELSE else_result_expression ]  
END
```

- ❑ CASE връща стойност (SQL е декларативен език, не става въпрос за flow control)
- ❑ Може да се използва в заявки на местата, където се очаква скаларна стойност

# Задача

За всяко студио - името и сумата от дължините на всички негови филми

```
SELECT STUDIOName, SUM(LENGTH)
FROM MOVIE
GROUP BY STUDIOName
```

```
SELECT DISTINCT STUDIOName, (SELECT SUM(LENGTH)
                                FROM MOVIE
                                WHERE STUDIOName = m.STUDIOName)
FROM MOVIE m
```