# SUMMARY OF SQL QUERIES

**SELECT** <attribute and function list>
**FROM** <table list>
[ **WHERE** <condition> ]
[ **GROUP BY** <grouping attribute(s)> ]
[ **HAVING** <group condition> ]
[ **ORDER BY** <attribute list> ];

1. Assemble all tables according to **From** clause (“,” means to use ×).
2. Keep only tuples matching **Where** clause.
3. Group into blocks based on **Group By** clause.
4. Keep only blocks matching **Having** clause.
5. Create one tuple for each block using **Select** clause.
6. Order resulting tuples according to **Order By** clause.

## Tables for the Example

Student

| ID | First | Last | Year |
|---|---|---|---|

Grade

| ID | Code | Mark | YearTaken |
|---|---|---|---|

Module

| Code | Title | Credits |
|---|---|---|

# A Final Example

- Examiners' reports
  - We want a list of students and their average mark
  - For first and second years the average is for that year
  - For finalists it is 40% of the second year plus 60% of the final year average.
- We want the results
  - Sorted by year then average mark (High to low) then last name, first name, and finally ID
  - To take into account the number of credits each module is worth
  - Produced by a single query

# We'll Need a UNION

- Finalists are treated differently
  - Write one query for the finalists
  - Write a second query for the first and second years
  - Use a UNION to join them together

```
<QUERY FOR FINALISTS>

UNION

<QUERY FOR OTHERS>
```

# We'll need to Join the Tables

- Both of the subqueries need information from all the tables
  - The student ID, name and year
  - The marks for each module and the year taken
  - The number of credits for each module
- This is a natural join operation
  - We could use a NATURAL JOIN statement, and hope that our version of SQL can do it
  - Safer to just use a WHERE clause

# The Query So Far

```
SELECT <some information>
  FROM Student, Module, Grade
 WHERE Student.ID = Grade.ID
   AND Module.Code = Grade.Code
   AND <student is in third year>

UNION

SELECT <some information>
  FROM Student, Module, Grade
 WHERE Student.ID = Grade.ID
   AND Module.Code = Grade.Code
   AND <student is in first or second year>
```

# Information for Finalists

- We need to retrieve
  - Compute average mark, weighted 40-60 across years 2 and 3
  - First year marks need to be ignored
  - The ID, Name, and Year are needed as they are used for ordering
- The average is hard
  - We don't have any statement to separate years 2 and 3 easily
  - We can exploit the fact that 40 = 20*2 and 60 = 20*3, so YearTaken and the weighting have a simple relationship

# Information for Finalists

```
SELECT Year, Student.ID, Last, First,
       SUM((20*YearTaken/100)*Mark*Credits)/120
          AS AverageMark
  FROM Student, Module, Grade
 WHERE Student.ID = Grade.ID
   AND Module.Code = Grade.Code
   AND YearTaken IN (2,3)
   AND Year = 3
 GROUP BY Year, Student.ID, First, Last
```

# Information for Other Students

- Other students are easier than finalists
  - We just need to average their marks where YearTaken and Year are the same
  - As before we need the ID, Name, and Year for ordering

# Information for Other Students

```
SELECT Year, Student.ID, Last, First,
       SUM(Mark*Credits)/120 AS AverageMark
  FROM Student, Module, Grade
 WHERE Student.ID = Grade.ID
   AND Module.Code = Grade.Code
   AND YearTaken = Year
   AND Year IN (1,2)
 GROUP BY Year, Student.ID, First, Last
```

# The Final Query

```
SELECT Year, Student.ID, Last, First,
       SUM((20*YearTaken/100)*Mark*Credits)/120 AS AverageMark
  FROM Student, Module, Grade
 WHERE Student.ID = Grade.ID AND Module.Code = Grade.Code
   AND YearTaken IN (2,3) AND Year = 3
 GROUP BY Year, Student.ID, First, Last

UNION

SELECT Year, Student.ID, Last, First,
        SUM(Mark*Credits)/120 AS AverageMark
   FROM Student, Module, Grade
  WHERE Student.ID = Grade.ID AND Module.Code = Grade.Code
    AND YearTaken = Year AND Year IN (1,2)
  GROUP BY Year, Student.ID, First, Last

ORDER BY Year desc, AverageMark desc, First, Last, ID
```

inventory(inventory_id, user_id, item_id)
item(item_id, name)
favs(fav_id, user_id, item_id)

We want a query to remove a given user's items, but only duplicate copies. However, do not remove any copies of an item if the user has it favourited.

1.

DELETE FROM inventory WHERE inventory_id IN <user's duplicate items, but not favourited>

2.

SELECT item_id FROM favs WHERE user_id = $user_id

This query will get the user's favourited items

3.

SELECT MIN(inventory_id) as inv_id, user_id, item_id
FROM inventory GROUP BY user_id, item_id

This query will get only one copy of the user's items

4.

```
SELECT inventory_id
FROM inventory
LEFT JOIN
(SELECT MIN(inventory_id) as inv_id, user_id, item_id
FROM inventory GROUP BY user_id, item_id) as KeepRows
ON inventory.inventory_id = KeepRows.inv_id
WHERE KeepRows.inv_id IS NULL AND inventory.user_id =
$user_id
```

In the left join, all duplicate items will have KeepRows.inv_id as null because of the left join

5. The final query

```
DELETE FROM inventory
WHERE inventory_id
IN
(SELECT inventory_id
FROM inventory
LEFT JOIN
(SELECT MIN(inventory_id) as inv_id, user_id, item_id
FROM inventory GROUP BY user_id, item_id) as KeepRows
ON inventory.inventory_id = KeepRows.inv_id
WHERE KeepRows.inv_id IS NULL AND inventory.user_id =
$user_id AND inventory.item_id NOT IN (SELECT item_id
FROM favs WHERE user_id = $user_id))
```