Aggregation and Grouping – Key Points

- aggregation operators apply an operation to a group of rows
  - e.g. COUNT, SUM, MAX
- grouping allows rows to be grouped together so that an aggregation operator can be applied to each group instead of (only) the table as a whole
  - GROUP BY to specify column(s) used for grouping
  - HAVING allows the selection of only certain groups
  - ORDER BY sorts the rows in the result
- be aware of how NULL is handled

```
SELECT Dno,COUNT(Ssn)
FROM EMPLOYEE
GROUP BY Dno

SELECT Dno
FROM EMPLOYEE
GROUP BY Dno
HAVING COUNT(Ssn) > 2
```

Aggregation

Aggregation operators apply an operation to a group of rows.

operator	notes
SUM(s)	operator is applied to the values in the column *s*
AVG(s)	
MIN(s)	
MAX(s)	
COUNT(s)	
COUNT(*)	computes the number of rows

Duplicates are included, unless DISTINCT is used.
- e.g. COUNT(DISTINCT s)

Aggregation

```
SELECT COUNT(*) AS numMovies
FROM MOVIES
WHERE year >= 2000

SELECT MIN(length) AS shortest,
       MAX(length) AS longest
FROM MOVIES
WHERE year = 2012

SELECT COUNT(DISTINCT year)
FROM ( MOVIES M JOIN STARSIN SI
          ON M.title=SI.movieTitle ) JOIN
       MOVIESTAR MS ON SI.starName=MS.name
WHERE MS.name = 'Harrison Ford'
```

What to Count

```
SELECT COUNT(*)
FROM SAILOR NATURAL JOIN RESERVATION
```
– counts the number of reservations

```
SELECT COUNT(Sid)
FROM SAILOR NATURAL JOIN RESERVATION
```
– counts the number of reservations (Sid is PK, so cannot be NULL)

```
SELECT COUNT(Sname)
FROM SAILOR NATURAL JOIN RESERVATION
```
– counts the number of reservations made by sailors whose Sname is not NULL

```
SELECT COUNT(DISTINCT Sid)
FROM SAILOR NATURAL JOIN RESERVATION
```
– counts the number of different sailors who have made reservations

```
SELECT COUNT(DISTINCT Sname)
FROM SAILOR NATURAL JOIN RESERVATION
```
– counts the number of different names of sailors who have made reservations

Grouping

Grouping allows rows to be grouped together so that an aggregation operator can be applied to each group instead of (only) the table as a whole.

```
SELECT studioName, AVG(length)
FROM MOVIES
GROUP BY studioName
```

– GROUP BY groups rows which have the same value for the specified attribute(s)
– aggregation operations in the SELECT clause are applied to each group
– other columns in the SELECT clause can only be those used for grouping

Grouping

- use the primary key when you want to uniquely identify things

```
SELECT deptname, COUNT(*)
FROM EMPLOYEE NATURAL JOIN DEPARTMENT
GROUP BY deptnum

SELECT deptname, COUNT(*)
FROM EMPLOYEE NATURAL JOIN DEPARTMENT
GROUP BY deptname
```

– in the second query, different departments which happen to have the same name will be combined into a single count – this is probably not what is desired

Grouping

```
SELECT studioName, AVG(length)
FROM MOVIES
WHERE year = 2000
GROUP BY studioName
```

– WHERE selects individual rows before grouping is applied

```
SELECT studioName, AVG(length)
FROM MOVIES
GROUP BY studioName
HAVING COUNT(*) > 2
```

– HAVING selects groups
– aggregation operators in HAVING apply to just the one group
– columns in HAVING must be aggregated or grouping columns (same rule as for SELECT)

Grouping

What order are the clauses in the following query applied? What does it do?

```
SELECT Rating, COUNT(*)
FROM SAILOR
WHERE Age > 30
GROUP BY Rating
HAVING COUNT(*) > 1
ORDER BY Rating
```

NULL

NULL is ignored in aggregation – it does not count towards SUM, etc.

- but COUNT(*) counts all rows, even if they contain NULLs

NULL is treated as an ordinary value when forming groups.

- groups may have NULL values in grouping attributes, and those will be separate groups from those without NULLs

The value of an aggregation computed on an empty bag is NULL, except for COUNT (which returns 0).