Database Systems

SQL: Data Manipulation (DML)
Aggregates/Summaries

SELECT - Aggregates

- SQL standard defines five aggregate functions:

- SUM(expr) sum of values in specified column
- AVG(expr) average of values in specified column
- MIN(expr) smallest value in specified column
- MAX(expr) largest value in specified column

- COUNT(expr) number of values in specified column
- COUNT(*) number of rows of query results

- Note that an expression can be used in place of a column name inside an aggregate function

SELECT - Aggregates

- Operates on a single table column, returns single value
- COUNT, MIN, and MAX apply to both numeric and non-numeric fields (character sets and collation order)
- SUM and AVG used on numeric fields only
- Use DISTINCT before column name to eliminate duplicates
  - No effect with MIN/MAX
  - May have effect on SUM/AVG
- Each function eliminates NULLs first and operates only on remaining non-null values, except COUNT(*)
- COUNT(*) counts all rows of a table, regardless of NULLs or duplicate values

Aggregates – NULLs Beware

- NULLs can cause subtle problems for SUM() and AVG()
- **SELECT (SUM(price) – SUM(cost)), SUM(price - cost) FROM titles;**

price	cost
15.00	10.00
25.00	20.00
120.00	NULL
80.00	70.00

SELECT - Aggregates

- Aggregate functions can be used ONLY in SELECT and HAVING clauses
- If SELECT includes an aggregate function and there is no GROUP BY clause, SELECT cannot reference a column outside an aggregate function
- This is illegal:

```
SELECT type,
  SUM(sales)
FROM titles;
```

This is OK:

```
SELECT SUM(sales)
--
FROM titles;
```


SELECT - Aggregates

- Aggregate functions can be used ONLY in SELECT and HAVING clauses
- If SELECT includes aggregate function(s) and references column(s) outside an aggregate function
  -> These columns must be in the GROUP BY
- This is illegal:

```
SELECT type,
    SUM(sales)
FROM titles;
```

This is OK:

```
SELECT type,
    SUM(sales)
FROM titles
GROUP BY type;
```

SELECT - Aggregates

- Aggregate functions can be used ONLY in SELECT and HAVING clauses
- Trying to find those titles with sales above the average sale
- This is illegal:

```
SELECT title_name
FROM titles
WHERE sales >
    AVG(sales);
```

This is OK:

```
SELECT title_name
FROM titles
WHERE sales >
   (SELECT AVG(sales)
    FROM titles);
```


COUNT(expr) and COUNT(*)

```
SELECT
    COUNT(title_id) AS "COUNT(title_id)",
    COUNT(price) AS "COUNT(price)",
    COUNT(*) AS "COUNT(*)"
  FROM titles;
```

```
COUNT(title_id)  COUNT(price)  COUNT(*)
---------------  ------------  --------
             13            12        13
```


Use of COUNT(DISTINCT)

```
SELECT
    COUNT(price) AS "COUNT(price)",
    SUM(price)   AS "SUM(price)",
    AVG(price)   AS "AVG(price)"
  FROM titles;
```

```
COUNT(price)  SUM(price)  AVG(price)
------------  ----------  ----------
          12      220.65     18.3875
```

```
SELECT
    COUNT(DISTINCT price)
      AS "COUNT(DISTINCT)",
    SUM(DISTINCT price)
      AS "SUM(DISTINCT)",
    AVG(DISTINCT price)
      AS "AVG(DISTINCT)"
  FROM titles;
```

```
COUNT(*)
--------
      13
```

```
COUNT(DISTINCT)  SUM(DISTINCT)  AVG(DISTINCT)
---------------  -------------  -------------
             10         187.71        18.7710
```

Use of DISTINCT

```
SELECT COUNT(au_id)
        AS "COUNT(au_id)"
  FROM title_authors;

SELECT DISTINCT COUNT(au_id)
        AS "DISTINCT COUNT(au_id)"
  FROM title_authors;

SELECT COUNT(DISTINCT au_id)
        AS "COUNT(DISTINCT au_id)"
  FROM title_authors;
```

```
COUNT(au_id)
------------
          17

DISTINCT COUNT(au_id)
---------------------
                   17

COUNT(DISTINCT au_id)
---------------------
                    6
```


Use of MIN, MAX

- Find minimum, maximum

```
SELECT
    MIN(price) AS "Min price",
    MAX(price) AS "Max price",
    MAX(price) - MIN(price) AS "Range"
 FROM titles;
```

```
Min price Max price Range
--------- --------- -----
     6.95     39.95 33.00
```


Use of AVG, SUM

- Find average, sum

```
SELECT AVG(sales) AS "Avg.Sale",
       SUM(sales) AS "Total Sales"
FROM titles
WHERE type = 'biography';
```

```
  Avg.Sale Total Sales
---------- -----------
537173.667     1611521
```

GROUP BY

- Use GROUP BY clause to get sub-totals
- If WHERE clause is used with GROUP BY
  - WHERE is applied first
  - Then groups are formed from rows satisfying WHERE conditions
- NULL
  - Two nulls considered equal for purpose of GROUP BY
- How many GROUPs with no GROUP BY?

GROUP BY

```
SELECT
    au_id,
    COUNT(*) AS "num_books"
  FROM title_authors
  GROUP BY au_id;
```

au_id	num_books
A01	3
A02	4
A03	2
A04	4
A05	1
A06	3

List the number of books each author wrote or co-wrote

GROUP BY

- SELECT and GROUP BY closely related:
  - All column names in SELECT list must appear in GROUP BY clause, unless column name is used in an aggregate function
  - Each item in SELECT list must be single-valued per group
  - What is wrong with the following?

```
SELECT type, pub_id, COUNT(*)
  FROM titles
  GROUP BY type;          --Illegal
```

Multiple Grouping Columns

- List the number of books of each type for each publisher

```
SELECT
    pub_id,
    type,
    COUNT(*) AS "COUNT(*)"
  FROM titles
  GROUP BY pub_id, type
  ORDER BY pub_id ASC, "COUNT(*)" DESC;
```

```
pub_id type        COUNT(*)
------ ---------- --------
P01    biography          3
P01    history            1
P02    computer           1
P03    history            2
P03    biography          1
P04    psychology         3
P04    children           2
```


GROUP BY without Aggregates

- The 2nd form is the recommended SELECT query.

```
SELECT type
  FROM titles
  GROUP BY type;

SELECT DISTINCT type
  FROM titles;
```

```
type
----------
biography
children
computer
history
psychology
```


Restrict Groupings – HAVING

- HAVING clause is designed for use with GROUP BY to restrict groups that appear in final result table
- Not to be confused with WHERE
  - WHERE filters individual rows from FROM
  - HAVING filters grouped rows from GROUP BY
  - Use WHERE wherever possible; not HAVING
- Column names in HAVING clause
  - Can reference any items in SELECT

Use of HAVING

- List the number of books written (or co-written) by each author who has written three or more books

```
SELECT
    au_id,
    COUNT(*) AS "num_books"
FROM title_authors
GROUP BY au_id
HAVING COUNT(*) >= 3;
```

```
au_id num_books
----- ---------
A01             3
A02             4
A04             4
A06             3
```


Use of HAVING

- List the number of titles and average revenue for the types with average revenue over $1 million.

```
SELECT
    type,
    COUNT(price) AS "COUNT(price)",
    AVG(price * sales) AS "AVG revenue"
FROM titles
GROUP BY type
HAVING AVG(price * sales) > 1000000;
```

```
type       COUNT(price) AVG revenue
---------- ------------ -----------
biography             3 12484878.00
computer              1  1025396.65
```


Use of HAVING

- List the number of books of each type for each publisher, for publishers with more than 1 title of a type.

```
SELECT
    pub_id,
    type,
    COUNT(*) AS "COUNT(*)"
FROM titles
GROUP BY pub_id, type
HAVING COUNT(*) > 1
ORDER BY pub_id ASC, "COUNT(*)" DESC;
```

```
pub_id type       COUNT(*)
------ ---------- --------
P01    biography         3
P03    history           2
P04    psychology        3
P04    children          2
```