SQL Aggregation and Grouping: Comprehensive Review

DSC 208R - Data Management for Analytics

Introduction to SQL Aggregation

Aggregation functions perform a calculation on a set of rows and return a single summary value.

Schemas Used in Examples

- **Movie** (name, year, genre)
- **ActedIN** (actorname, moviename)
- **Actor** (name, age)

Data for Examples

Movie Table:

Name	Year	Genre
Apocalypse Now	1979	War
The God Father	1972	Crime
Planet Earth II	2016	Nature Documentary

ActedIN Table:

Actorname	Moviename
Marlon Brando	Apocalypse Now
Al Pacino	The God Father
Marlon Brando	The God Father

Actor Table:

Name	Age
Marlon Brando	80
Al Pacino	82
De Niro	79

SQL Aggregation Functions

Common aggregation functions include:

- `COUNT(X)`: Counts the number of non-NULL values in column X.
- `SUM(X)`: Computes the sum of values in column X.
- `AVG(X)`: Computes the average of values in column X.
- `MIN(X)`: Finds the minimum value in column X.
- `MAX(X)`: Finds the maximum value in column X.

All these functions ignore NULL values.

'COUNT'

Query: How many movies are there in the table?

```
SELECT COUNT(*) FROM Movie;
```

Result:

COUNT(*)
3

‘COUNT(DISTINCT X)‘

Query: Count how many distinct genres are there.

```
SELECT COUNT(DISTINCT genre) FROM Movie;
```

Result:

COUNT(DISTINCT genre)
3

‘SUM‘

Query: What is the total length of all movies? (Assuming a ‘length‘ column exists)

```
SELECT SUM(length) FROM Movie;
```

Explanation: This query calculates the sum of all values in the ‘length‘ column of the ‘Movie‘ table.

‘AVG‘

Query: What is the average length of all movies?

```
SELECT AVG(length) FROM Movie;
```

Explanation: This query calculates the average of all values in the ‘length‘ column of the ‘Movie‘ table.

‘MIN‘ and ‘MAX‘

Query: What are the minimum and maximum lengths of all movies?

```
SELECT MIN(length), MAX(length) FROM Movie;
```

Explanation: This query retrieves the smallest value using ‘MIN()‘ and the largest value using ‘MAX()‘ from the ‘length‘ column of the ‘Movie‘ table.

Grouping in SQL: ‘GROUP BY‘ Clause

The ‘GROUP BY‘ clause groups rows that have the same values in specified columns into summary rows, like ”count the number of customers in each country”. It is used with aggregation functions to perform calculations on each group.

Execution Order:

1. `FROM` clause: Identifies the tables involved.
2. `WHERE` clause: Filters rows based on specified conditions.
3. `GROUP BY` clause: Groups the filtered rows into sets based on common values in the specified column(s).
4. Aggregates: Computes aggregate functions (e.g., ‘COUNT‘, ‘SUM‘) for each group.
5. `SELECT` clause: Selects the columns and aggregate results to display.

‘GROUP BY‘ Example 1

Query: How many movies in each genre?

```
SELECT genre, COUNT(*)
FROM Movie
GROUP BY genre;
```

Result:

genre	COUNT(*)
War	1
Crime	1
Nature Documentary	1

Explanation: This query groups the ‘Movie‘ table by ‘genre‘ and then counts the number of movies within each unique genre.

‘GROUP BY‘ Example 2

Query: How many movies produced in each year?

```
SELECT year, COUNT(name)
FROM Movie
GROUP BY year;
```

Result:

year	COUNT(name)
1979	1
1972	1
2016	1

Explanation: This query groups the ‘Movie‘ table by ‘year‘ and then counts the number of movie names within each production year.

Filtering Groups: ‘HAVING‘ Clause

The ‘HAVING‘ clause is used to filter groups based on conditions applied to aggregate values. It operates on the results of the ‘GROUP BY‘ clause.

Execution Order with ‘HAVING‘:

1. `FROM`
2. `WHERE`
3. `GROUP BY`
4. `HAVING` (filters groups)
5. Aggregates (re-computes if needed for `SELECT`)
6. `SELECT`

‘HAVING‘ Example

Query: Find genres with more than 10 movies.

```
SELECT genre, COUNT(*)
FROM Movie
GROUP BY genre
HAVING COUNT(*) > 10;
```

Explanation: This query first groups movies by ‘genre‘ and counts them. Then, the ‘HAVING‘ clause filters these groups, keeping only those genres where the count of movies is greater than 10.