MSDS 597

Lecture 3

CRUD Operations

All databases need to be able to handle 4 basic operations, abbreviated CRUD

- Create
  - CREATE: Creates a container for data to be stored (e.g. a table)
  - INSERT: Put data into that container
  - COPY: Bulk data loading
- Read
  - SELECT: Primary way of retrieving data from a database
- Update
  - UPDATE: Change data within a table
  - ALTER: Change the structure of a table
- Delete
  - DROP: Remove a database object
  - DELETE: Remove data from a table

GROUP BY

- Aggregate data into specified groups
- Scans multiple rows and collapses them into a single aggregate value
- How does it work?
  - Combine rows that have the same value for the grouping variable
  - Use an aggregate function (e.g. SUM, AVG, COUNT …) to collapse/summarize the other columns of interest into a single value for the group
  - In the SELECT, all columns are either 1) part of the group or 2) being aggregated.
  - Applied after the WHERE clause
    - Rows are filtered out first
    - Then aggregated
  - The result will always have **one** row per group
  - Groups can consist of one or more columns

GROUP BY (Single Column) - Illustrated Example

year	make	model	sales_mm
2020	Toyota	Prius	5.5
2020	Toyota	Rav 4	4.5
2020	Honda	Civic	8.2
2021	Toyota	Prius	6.3

```
SELECT
    year,
    SUM(sales_mm) AS total_sales_mm
FROM sales
GROUP BY year
ORDER BY year;
```

GROUP BY 1
ORDER BY 1
is also OK

year	total_sales_mm
2020	18.2
2021	6.3

GROUP BY (Multiple Columns) - Illustrated Example

year	make	model	sales_mm
2020	Toyota	Prius	5.5
2020	Toyota	Rav 4	4.5
2020	Honda	Civic	8.2
2021	Toyota	Prius	6.3

```
SELECT
    year,
    make,
    SUM(sales_mm) AS total_sales_mm
FROM sales
GROUP BY year, make
ORDER BY year, make;
```

year	make	total_sales_mm
2020	Toyota	10.0
2020	Honda	8.2
2021	Toyota	6.3

GROUP BY 1, 2
ORDER BY 1, 2
is also OK

Common Aggregation Functions

COUNT

- COUNT(1) or COUNT(*): Counts the number of rows
- COUNT(col_name): Counts the number of Non-NULL values in col_name
- COUNT(DISTINCT col_name):
    - Counts the number of distinct Non-NULL values in col_name

SUM

- SUM(col_name): Sums the values in col_name

AVG

- AVG(col_name): Averages the values in col_name

MAX/MIN

- MAX(col_name): Return the maximum value in col_name
- MIN(col_name): Return the minimum value in col_name

NULL values and Aggregations

If one of the grouping column values is NULL ...

- NULL is treated as a group value and all NULLs are grouped together

If NULLs are aggregated

- SUM, MAX, MIN, COUNT(col_name), COUNT(DISTINCT col_name)
  - NULLs are ignored
- COUNT(1) or COUNT(*)
  - NULLs are counted

CASE statements and Aggregations

CASE statements can be used to create new groups

- Transform existing columns/dimensions into new ones to categorize the data in new ways

CASE statements can be used inside an aggregation to only selectively aggregate

- Useful for calculating % of total
  - SUM(CASE WHEN brand = ‘Toyota’ THEN sales ELSE 0 END) * 100.0 / SUM(sales)

Multiple levels of aggregation

Sometimes you need an aggregation on top of another aggregation.

Example: What is the highest average selling price among car models?

Solution: Use a subquery to do the first aggregation (average price in the example above). Then aggregate the result of the subquery in the outer query with a second aggregation (maximum of the average price in the example above)

model	selling_price
Prius	10000
Prius	15000
Accord	13000
Accord	17000

AVG →

model	avg_selling_price
Prius	12500
Accord	15000

MAX →

max_avg_selling_price
15000

HAVING clause

HAVING is a post-aggregation filter

- This is in contrast to WHERE, which is a pre-aggregation filter
- In other words, WHERE filters rows and HAVING filters groups
- Typically, use an aggregation-based filter in HAVING
    - e.g. HAVING COUNT(1) > 1

DATE data types

Dates are the least standardized part of SQL

Date data types:

- DATE: Stores a date e.g. 2021-01-01
- TIME: Stores a time
- TIMESTAMP: Stores a date and a time
  - e.g. 2021-01-01 01:59:42.892321 +00:00
  - Sidenote: You may see “timestamps” in data that look like this 1636274895
    - This is unixtime. It represents the number of elapsed seconds from 1970-01-01 and is used to make time-based calculations more convenient
- INTERVAL: Stores a time interval e.g. 1 year or 3 seconds

DATE functions

NOW() : Returns the current timestamp

CURRENT_DATE : Returns the current date

CURRENT_TIME : Returns the current time

DATE_PART(part_string ,date) : Extracts a particular part of the date or time

DATE_TRUNC(granularity, date) : Truncates a date or timestamp to a certain precision

Intervals are used to add or subtract time information e.g SELECT ‘2021-01-01’::DATE + ‘1 day’::INTERVAL