

ISLAMIC UNIVERSITY OF TECHNOLOGY



RELATIONAL DATABASE MANAGEMENT
SYSTEM LAB

CSE 4508

Lab 4: Aggregation, Grouping, ROLLUP, and CUBE in SQL

Author:

Ahmed M. S. Albreem (210041258)

October 15, 2024

Task 1: Total Kilometers, Fuel, and Passengers per Bus, Grouped by Year, Month, and Day

Task Explanation

In this task, we calculate the total kilometers traveled, fuel consumed, and passenger count for each bus, grouped by `year`, `month`, and `day`. This query aims to provide a comprehensive overview of the operational metrics of each bus over time.

To achieve this, we utilize the following SQL features:

- **`bus_id`**: This field serves as the unique identifier for each bus in the dataset, allowing us to distinguish between different vehicles.
- **`EXTRACT(YEAR FROM trip_date)`, `EXTRACT(MONTH FROM trip_date)`, `EXTRACT(DAY FROM trip_date)`**: These functions are utilized to retrieve the respective year, month, and day from the `trip_date` column. By extracting these components, we can perform time-based analysis and group the data accordingly.
- **`SUM(km_travelled)`, `SUM(fuel)`, `SUM(passengers)`**: These aggregate functions compute the total values for kilometers traveled, fuel consumed, and passenger counts. Each function aggregates the values for its respective column, providing a summary of the total metrics associated with each bus on a daily basis.

The complete SQL query for this task is as follows:

```
1 SELECT
2     b.bus_model ,
3     YEAR(t.trip_date) AS trip_year ,
4     MONTH(t.trip_date) AS trip_month ,
5     DAY(t.trip_date) AS trip_day ,
6     SUM(t.kilometers_traveled) AS total_km ,
7     SUM(t.fuel_consumption) AS total_fuel_consumed ,
8     SUM(t.passengers_count) AS total_passengers
9 FROM
10    trip t
11 JOIN bus b ON t.bus_id = b.bus_id
```

The screenshot shows a database result grid with the following columns: bus_model, trip_year, trip_month, trip_day, total_km, total_fuel_consumed, and total_passengers. The data is as follows:

| bus_model | trip_year | trip_month | trip_day | total_km | total_fuel_consumed | total_passengers |
|-----------|-----------|------------|----------|----------|---------------------|------------------|
| Model A | 2023 | 10 | 1 | 100 | 50 | 20 |
| Model A | 2023 | 10 | 3 | 150 | 60 | 25 |
| Model B | 2023 | 10 | 2 | 200 | 75 | 30 |
| Model B | 2023 | 10 | 1 | 220 | 90 | 50 |
| Model C | 2023 | 11 | 1 | 300 | 100 | 40 |
| Model C | 2023 | 10 | 2 | 180 | 70 | 45 |
| Model D | 2023 | 11 | 2 | 250 | 80 | 35 |
| Model D | 2023 | 10 | 3 | 320 | 120 | 55 |

Figure 1: Task 1

```

12 GROUP BY
13     b.bus_model, trip_year, trip_month, trip_day;

```

Listing 1: SQL Query for Task 1

Task 2: Using ROLLUP for Bus Model, Year, and Month

This task uses the ROLLUP function to generate subtotals for bus models by year and month, as well as grand totals.

Explanation

The ROLLUP function allows for hierarchical subtotal generation in the results. It simplifies the process of summarizing data across different levels of aggregation.

```

1 SELECT
2     b.bus_model,
3     YEAR(t.trip_date) AS trip_year,
4     MONTH(t.trip_date) AS trip_month,
5     SUM(t.kilometers_traveled) AS total_km,
6     SUM(t.fuel_consumption) AS total_fuel_consumed,
7     SUM(t.passengers_count) AS total_passengers
8 FROM
9     trip t

```

| | bus_model | trip_year | trip_month | total_km | total_fuel_consumed | total_passengers |
|---|-----------|-----------|------------|----------|---------------------|------------------|
| ▶ | Model A | 2023 | 10 | 250 | 110 | 45 |
| | Model A | 2023 | NULL | 250 | 110 | 45 |
| | Model A | NULL | NULL | 250 | 110 | 45 |
| | Model B | 2023 | 10 | 420 | 165 | 80 |
| | Model B | 2023 | NULL | 420 | 165 | 80 |
| | Model B | NULL | NULL | 420 | 165 | 80 |
| | Model C | 2023 | 10 | 180 | 70 | 45 |
| | Model C | 2023 | 11 | 300 | 100 | 40 |
| | Model C | 2023 | NULL | 480 | 170 | 85 |
| | Model C | NULL | NULL | 480 | 170 | 85 |
| | Model D | 2023 | 10 | 320 | 120 | 55 |
| | Model D | 2023 | 11 | 250 | 80 | 35 |
| | Model D | 2023 | NULL | 570 | 200 | 90 |
| | Model D | NULL | NULL | 570 | 200 | 90 |
| | NULL | NULL | NULL | 1720 | 645 | 300 |

Figure 2: Task 2

```
10 JOIN bus b ON t.bus_id = b.bus_id
11 GROUP BY b.bus_model, trip_year, trip_month WITH ROLLUP;
```

Listing 2: SQL Query for Task 2

Task 3: Using CUBE for Bus Model, Depot, and Year

In this task, we utilize the CUBE function to calculate all possible combinations of subtotals for bus model, depot, and year.

The screenshot shows a database query results grid with the following columns: bus_model, depot_name, trip_year, total_km, total_fuel_consumed, and total_passengers. The data is grouped by bus_model and depot_name, with trip_year being the detailed dimension. The results show various bus models (Model A, Model B, Model C, Model D) operating from different depots (Depot 1, Depot 2, Depot 3) across different years (2023). Metrics include total kilometers traveled, total fuel consumed, and total passengers carried.

| | bus_model | depot_name | trip_year | total_km | total_fuel_consumed | total_passengers |
|----|-----------|------------|-----------|----------|---------------------|------------------|
| 1 | Model B | Depot 1 | 2023 | 220 | 90 | 50 |
| 2 | Model B | Depot 2 | 2023 | 200 | 75 | 30 |
| 3 | Model C | Depot 2 | 2023 | 480 | 170 | 85 |
| 4 | Model D | Depot 3 | 2023 | 570 | 200 | 90 |
| 5 | Model A | Depot 1 | NULL | 250 | 110 | 45 |
| 6 | Model B | Depot 1 | NULL | 220 | 90 | 50 |
| 7 | Model B | Depot 2 | NULL | 200 | 75 | 30 |
| 8 | Model C | Depot 2 | NULL | 480 | 170 | 85 |
| 9 | Model D | Depot 3 | NULL | 570 | 200 | 90 |
| 10 | Model A | NULL | 2023 | 250 | 110 | 45 |
| 11 | Model B | NULL | 2023 | 420 | 165 | 80 |
| 12 | Model C | NULL | 2023 | 480 | 170 | 85 |
| 13 | Model D | NULL | 2023 | 570 | 200 | 90 |
| 14 | NULL | Depot 1 | 2023 | 470 | 200 | 95 |
| 15 | NULL | Depot 2 | 2023 | 680 | 245 | 115 |
| 16 | NULL | Depot 3 | 2023 | 570 | 200 | 90 |
| 17 | Model A | NULL | NULL | 250 | 110 | 45 |
| 18 | Model B | NULL | NULL | 420 | 165 | 80 |
| 19 | Model C | NULL | NULL | 480 | 170 | 85 |
| 20 | Model D | NULL | NULL | 570 | 200 | 90 |
| 21 | NULL | Depot 1 | NULL | 470 | 200 | 95 |
| 22 | NULL | Depot 2 | NULL | 680 | 245 | 115 |
| 23 | NULL | Depot 3 | NULL | 570 | 200 | 90 |
| 24 | NULL | NULL | 2023 | 1720 | 645 | 300 |
| 25 | NULL | NULL | NULL | 1720 | 645 | 300 |

Figure 3: Task 3

Explanation

The CUBE function provides a multidimensional view of the data, allowing us to summarize metrics across various dimensions, thus enabling comprehensive analysis.

```

1 SELECT
2     b.bus_model ,
3     d.depot_name ,
4     YEAR(t.trip_date) AS trip_year ,
5     SUM(t.kilometers_traveled) AS total_km ,
6     SUM(t.fuel_consumption) AS total_fuel_consumed ,
7     SUM(t.passengers_count) AS total_passengers
8 FROM
9     trip t
10 JOIN bus b ON t.bus_id = b.bus_id
11 JOIN depot d ON t.depot_id = d.depot_id
12 GROUP BY b.bus_model , d.depot_name , trip_year WITH CUBE;

```

Listing 3: SQL Query for Task 3

Task 4: Handling Rollup Data in MySQL

This task addresses the handling of rollup data in MySQL, noting that MySQL does not support the GROUPING() function.

Explanation

MySQL's WITH ROLLUP generates subtotals and grand totals. The GROUPING() function, which exists in some SQL dialects (like Oracle), is not directly available in MySQL. Instead, you typically identify subtotal rows by checking for NULL values in the grouping columns within your application logic.

```
1 SELECT
2     b.bus_model ,
3     YEAR(t.trip_date) AS trip_year ,
4     MONTH(t.trip_date) AS trip_month ,
5     SUM(t.kilometers_traveled) AS total_km ,
6     SUM(t.fuel_consumption) AS total_fuel_consumed ,
7     SUM(t.passengers_count) AS total_passengers
8 FROM
9     trip t
10 JOIN bus b ON t.bus_id = b.bus_id
11 GROUP BY b.bus_model , trip_year , trip_month WITH ROLLUP;
```

Listing 4: SQL Query for Task 4 (MySQL adaptation)

The screenshot shows a database result grid titled 'Result Grid' with various export options at the top. The grid displays data for four bus models (Model A, Model B, Model C, Model D) across different years (2023) and months (10, 11). The columns are labeled: bus_model, trip_year, trip_month, total_km, total_fuel_consumed, and total_passengers. The data is summarized as follows:

| bus_model | trip_year | trip_month | total_km | total_fuel_consumed | total_passengers |
|-----------|-----------|------------|----------|---------------------|------------------|
| Model A | 2023 | 10 | 250 | 110 | 45 |
| Model A | 2023 | NULL | 250 | 110 | 45 |
| Model A | NULL | NULL | 250 | 110 | 45 |
| Model B | 2023 | 10 | 420 | 165 | 80 |
| Model B | 2023 | NULL | 420 | 165 | 80 |
| Model B | NULL | NULL | 420 | 165 | 80 |
| Model C | 2023 | 10 | 180 | 70 | 45 |
| Model C | 2023 | 11 | 300 | 100 | 40 |
| Model C | 2023 | NULL | 480 | 170 | 85 |
| Model C | NULL | NULL | 480 | 170 | 85 |
| Model D | 2023 | 10 | 320 | 120 | 55 |
| Model D | 2023 | 11 | 250 | 80 | 35 |
| Model D | 2023 | NULL | 570 | 200 | 90 |
| Model D | NULL | NULL | 570 | 200 | 90 |
| NULL | NULL | NULL | 1720 | 645 | 300 |

Figure 4: Task 4

Task 5: Total Kilometers, Fuel, and Passengers Grouped by Bus Model, Depot, and Year

In this final task, we calculate the total kilometers traveled, fuel consumed, and passenger count grouped by bus model, depot, and year.

Explanation

This task focuses on aggregating the data by key identifiers, enabling the analysis of operational efficiency across different depots and bus models.

```

1 SELECT
2     b.bus_model ,
3     d.depot_name ,
4     YEAR(t.trip_date) AS trip_year ,
5     SUM(t.kilometers_traveled) AS total_km ,
6     SUM(t.fuel_consumption) AS total_fuel_consumed ,
7     SUM(t.passengers_count) AS total_passengers
8 FROM
9     trip t
10 JOIN bus b ON t.bus_id = b.bus_id
11 JOIN depot d ON t.depot_id = d.depot_id

```

The screenshot shows a 'Result Grid' window with the following details:

- Toolbar:** Includes 'Result Grid' button, 'Filter Rows:' dropdown, 'Export:' button, and 'Wrap Cell Content:' button.
- Table Headers:** bus_model, depot_name, trip_year, total_km, total_fuel_consumed, total_passengers.
- Data Rows:**
 - Model A, Depot 1, 2023, 250, 110, 45
 - Model B, Depot 1, 2023, 220, 90, 50
 - Model B, Depot 2, 2023, 200, 75, 30
 - Model C, Depot 2, 2023, 480, 170, 85
 - Model D, Depot 3, 2023, 570, 200, 90
- Bottom Right:** A small button labeled '200'.

Figure 5: Task 5

```
12 GROUP BY b.bus_model, d.depot_name, trip_year;
```

Listing 5: SQL Query for Task 5