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

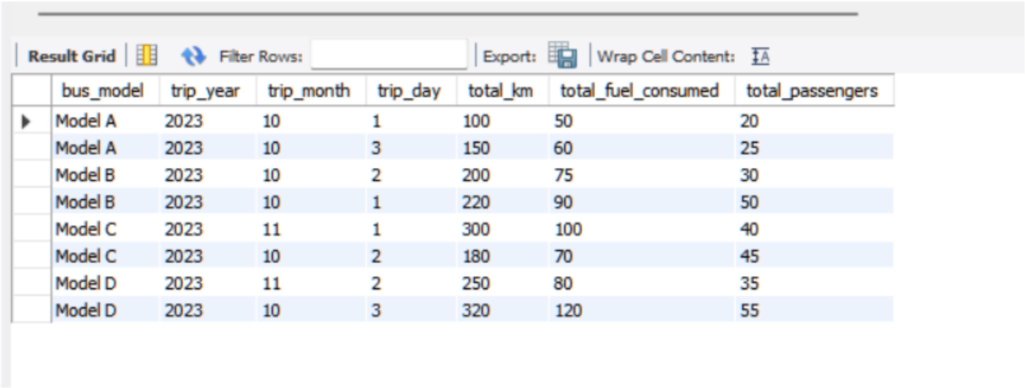
In this task, we calculate the total kilometers traveled, fuel consumed, and passenger count for each bus, grouped by **year**, **month**, and **day**. This query aims to provide a comprehensive overview of the operational metrics of each bus over time.

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:

```sql
SELECT
    b.bus_model,
    YEAR(t.trip_date) AS trip_year,
    MONTH(t.trip_date) AS trip_month,
    DAY(t.trip_date) AS trip_day,
    SUM(t.kilometers_traveled) AS total_km,
    SUM(t.fuel_consumption) AS total_fuel_consumed,
    SUM(t.passengers_count) AS total_passengers
FROM
    trip t
JOIN bus b ON t.bus_id = b.bus_id
```



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

```sql
GROUP BY
    b.bus_model, trip_year, trip_month, trip_day;
```

Listing 1: SQL Query for Task 1

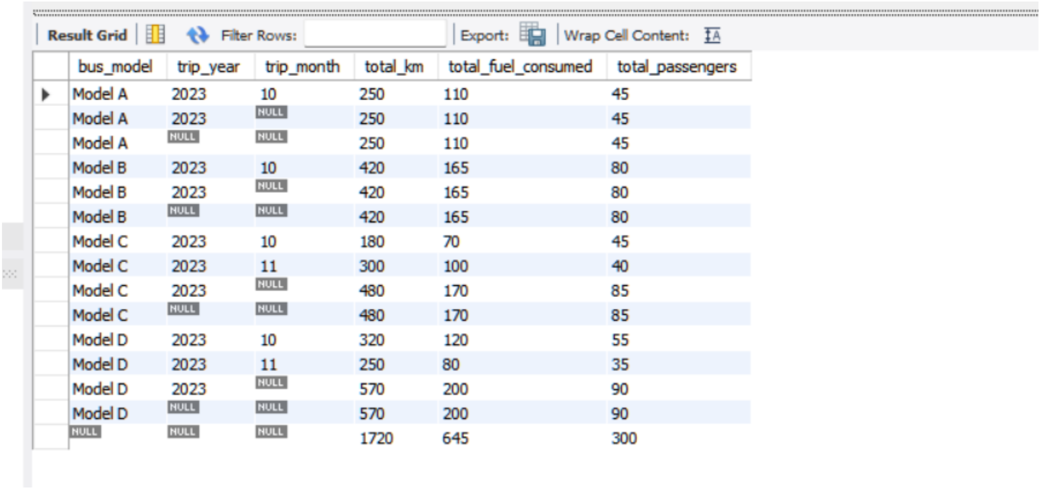
Task 2: Using ROLLUP for Bus Model, Year, and Month

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.

```sql
SELECT
    b.bus_model,
    YEAR(t.trip_date) AS trip_year,
    MONTH(t.trip_date) AS trip_month,
    SUM(t.kilometers_traveled) AS total_km,
    SUM(t.fuel_consumption) AS total_fuel_consumed,
    SUM(t.passengers_count) AS total_passengers
FROM
    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

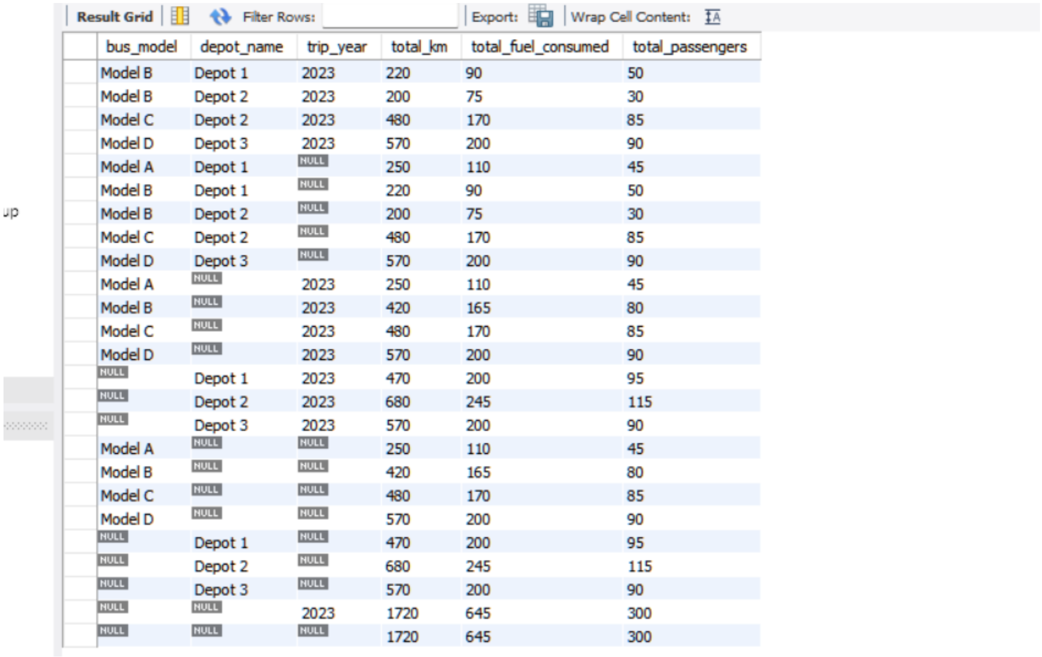
Figure 2: Task 2

```sql
JOIN bus b ON t.bus_id = b.bus_id
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.



bus_model	depot_name	trip_year	total_km	total_fuel_consumed	total_passengers
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
Model A	Depot 1	NULL	250	110	45
Model B	Depot 1	NULL	220	90	50
Model B	Depot 2	NULL	200	75	30
Model C	Depot 2	NULL	480	170	85
Model D	Depot 3	NULL	570	200	90
Model A	NULL	2023	250	110	45
Model B	NULL	2023	420	165	80
Model C	NULL	2023	480	170	85
Model D	NULL	2023	570	200	90
NULL	Depot 1	2023	470	200	95
NULL	Depot 2	2023	680	245	115
NULL	Depot 3	2023	570	200	90
Model A	NULL	NULL	250	110	45
Model B	NULL	NULL	420	165	80
Model C	NULL	NULL	480	170	85
Model D	NULL	NULL	570	200	90
NULL	Depot 1	NULL	470	200	95
NULL	Depot 2	NULL	680	245	115
NULL	Depot 3	NULL	570	200	90
NULL	NULL	2023	1720	645	300
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.

```sql
SELECT
    b.bus_model,
    d.depot_name,
    YEAR(t.trip_date) AS trip_year,
    SUM(t.kilometers_traveled) AS total_km,
    SUM(t.fuel_consumption) AS total_fuel_consumed,
    SUM(t.passengers_count) AS total_passengers
FROM
    trip t
JOIN bus b ON t.bus_id = b.bus_id
JOIN depot d ON t.depot_id = d.depot_id
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

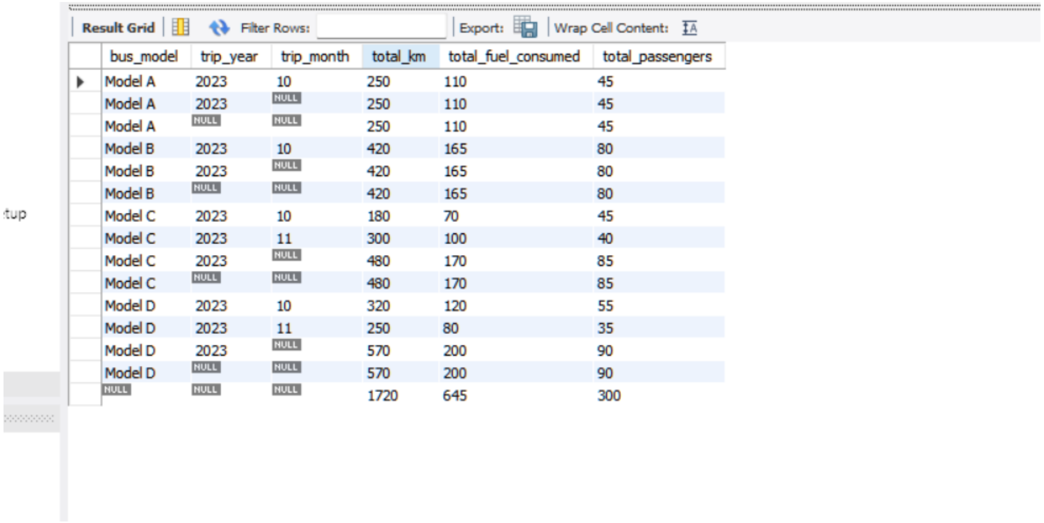
This task addresses the handling of rollup data in MySQL, noting that MySQL does not support the GROUPING() function.

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.

```sql
SELECT
    b.bus_model,
    YEAR(t.trip_date) AS trip_year,
    MONTH(t.trip_date) AS trip_month,
    SUM(t.kilometers_traveled) AS total_km,
    SUM(t.fuel_consumption) AS total_fuel_consumed,
    SUM(t.passengers_count) AS total_passengers
FROM
    trip t
JOIN bus b ON t.bus_id = b.bus_id
GROUP BY b.bus_model, trip_year, trip_month WITH ROLLUP;
```

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



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

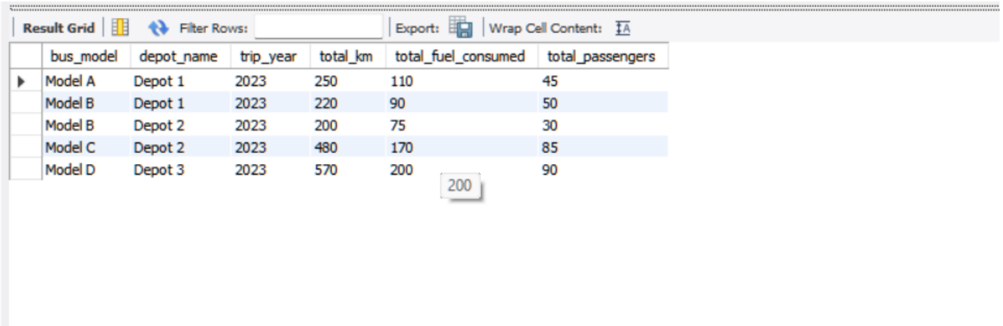
Task 5: Total Kilometers, Fuel, and Passengers Grouped by Bus Model, Depot, and Year

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.

```sql
SELECT
    b.bus_model,
    d.depot_name,
    YEAR(t.trip_date) AS trip_year,
    SUM(t.kilometers_traveled) AS total_km,
    SUM(t.fuel_consumption) AS total_fuel_consumed,
    SUM(t.passengers_count) AS total_passengers
FROM
    trip t
JOIN bus b ON t.bus_id = b.bus_id
JOIN depot d ON t.depot_id = d.depot_id
```



bus_model	depot_name	trip_year	total_km	total_fuel_consumed	total_passengers
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

Figure 5: Task 5

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

Listing 5: SQL Query for Task 5