SQL Code Explanation

1. Find top 10 highest revenue generating products

This SQL query fetches the top 10 products with the highest total sales from the `retail_orders` table, displaying each `product_id` along with its corresponding total sales value.

- **`SELECT product_id, sum(sell_price) AS sales`**:
  Selects the `product_id` and calculates the total sales for each product by summing the `sell_price` values.
- **`FROM retail_orders`**:
  Specifies the `retail_orders` table as the data source.
- **`GROUP BY product_id`**:
  Groups the results by `product_id`, so the sum of sales is calculated for each individual product.
- **`ORDER BY sales DESC`**:
  Orders the results in descending order based on the `sales` (total sales value).
- **`LIMIT 10`**:
  Limits the result to the top 10 products with the highest sales.

```sql
select product_id, sum(sell_price) as sales
from retail_orders
group by product_id
order by sales desc
limit 10;
```

1. Select top 5 highest selling products in each region

This SQL query fetches the top 5 products by total sales within each region from the `retail_orders` table, displaying the `region`, `product_id`, and corresponding sales for each of the top products.

- **`WITH cte AS (...)`**:
  A Common Table Expression (CTE) is used to calculate the total sales (`sum(sell_price)`) for each combination of `region` and `product_id` in the `retail_orders` table, grouped by `region` and `product_id`.
- **`SELECT * FROM (...) A`**:
  Selects all columns from the result of the inner query (the CTE).
- **`ROW_NUMBER() OVER (PARTITION BY region ORDER BY sales DESC) AS rn`**:
  For each region, assigns a row number to the products, ordered by sales in descending order.
- **`WHERE rn <= 5`**:
  Filters the result to include only the top 5 products (`rn <= 5`) by sales within each region.

```sql
with cte as (
select region, product_id, sum(sell_price) as sales
from retail_orders
group by region, product_id)
select * from (
select *
, row_number() over (partition by region order by sales desc) as rn
from cte) A
where rn <= 5;
```

1. Find month over month growth comparison for 2022 and 2023 sales eg : jan 2022 vs jan 2023

This SQL query compares the monthly sales for 2022 and 2023, presenting the sales data for each month in both years, and orders the results by month.

- **`WITH cte AS (...)`**:
  The Common Table Expression (CTE) calculates the total sales (`sum(sell_price)`) for each month and year by extracting the year and month from `order_date` and grouping the results accordingly.
- **`SELECT order_month, ... FROM cte`**:
  In the main query, it selects the `order_month` and calculates the total sales for each month for the years 2022 and 2023.
- **`SUM(CASE WHEN order_year=2022 THEN sales ELSE 0 END) AS sales_2022`**:
  Uses a `CASE` statement to sum sales for the year 2022. If the `order_year` is 2022, the sales are included; otherwise, it's 0.
- **`SUM(CASE WHEN order_year=2023 THEN sales ELSE 0 END) AS sales_2023`**:
  Similarly, it sums sales for 2023.
- **`GROUP BY order_month`**:
  Groups the data by month so that the sales for each month in both years are aggregated.
- **`ORDER BY order_month`**:
  Orders the results by the month.

```sql
with cte as (
select year(order_date) as order_year, month(order_date) as order_month,
sum(sell_price) as sales
from retail_orders
group by year(order_date), month(order_date)
-- order by year(order_date), month(order_date)
)
select order_month
, sum(case when order_year=2022 then sales else 0 end) as sales_2022
, sum(case when order_year=2023 then sales else 0 end) as sales_2023
from cte
group by order_month
order by order_month;
```

1. For each category which month had highest sales

This SQL query identifies the month with the highest sales for each product category, displaying the `category`, `order_year_month`, and the corresponding sales figures.

- **`WITH cte AS (...)`**:
  A Common Table Expression (CTE) calculates the total sales (`sum(sell_price)`) for each `category` and `order_year_month` (formatted as `yyyyMM` from `order_date`).
- **`SELECT * FROM (...) a`**:
  In the main query, it selects all columns from the CTE.
- **`ROW_NUMBER() OVER (PARTITION BY category ORDER BY sales DESC) AS rn`**:
  For each `category`, it assigns a row number based on sales in descending order, so the highest sales month gets `rn = 1`.
- **`WHERE rn = 1`**:
  Filters the result to only include the month with the highest sales for each category (i.e., the row where `rn = 1`).

```sql
with cte as (
select category, format(order_date,'yyyyMM') as order_year_month
, sum(sell_price) as sales
from retail_orders
group by category, format(order_date,'yyyyMM')
-- order by category, format(order_date,'yyyyMM')
)
select * from (
select *,
row_number() over(partition by category order by sales desc) as rn
from cte
) a
where rn = 1;
```

1. Which sub category had highest growth by profit in 2023 compare to 2022

This SQL query identifies the subcategory with the greatest sales growth between 2022 and 2023, displaying the sub_category, sales for both years, and the difference in sales.

- **`WITH cte AS (...)`**:
  The first CTE calculates the total sales (`sum(sell_price)`) for each `sub_category` and year by grouping the data by `sub_category` and `order_year`.
- **`WITH cte2 AS (...)`**:
  The second CTE aggregates the sales for each `sub_category` separately for 2022 and 2023 using `CASE` statements. It calculates `sales_2022` and `sales_2023` for each subcategory.
- **`SELECT * , (sales_2023 - sales_2022)`**:
  Selects all columns from `cte2` and computes the difference in sales between 2023 and 2022 for each `sub_category`.
- **`ORDER BY (sales_2023 - sales_2022) DESC`**:
  Orders the results by the sales difference in descending order, showing the subcategory with the largest increase in sales at the top.
- **`LIMIT 1`**:
  Limits the result to only the subcategory with the highest sales difference.

```
with cte as (
select sub_category, year(order_date) as order_year,
sum(sell_price) as sales
from retail_orders
group by sub_category, year(order_date)
-- order by year(order_date)
)
, cte2 as (
select sub_category
, sum(case when order_year=2022 then sales else 0 end) as sales_2022
, sum(case when order_year=2023 then sales else 0 end) as sales_2023
from cte
group by sub_category
)
select *
, (sales_2023-sales_2022)
from cte2
order by (sales_2023-sales_2022) desc
limit 1;
```