Request_1: Provide a list of products with the base price greater than 500 and that are featured in promo type of BOGOF (By One Get One Free) this information will help us identify high value products that are currently being heavily discounted which can be useful for evaluating our pricing and promotion strategies.

```
USE retail_events_db;

SELECT

p.product_code,

p.product_name,

f.base_price,

f.promo_type

FROM dim_products p

JOIN fact_events f

ON p.product_code = f.product_code

WHERE f.base_price > 500

  AND f.promo_type = 'BOGOF';
```

Request_2 : Generate a report that provides an overview of number of stores in each city. the results will be sorted in descending order of store counts allowing us to identify the cities with the highest store presence. The report includes two essential fields city and store count, which will assist in optimising our retail operation.

```
USE retail_events_db;

SELECT city,

COUNT(store_id) as Stores_Count

FROM dim_stores

GROUP BY city

ORDER BY Stores_count DESC;
```

Request_3 Generate a report that displays each campaign along with the total revenue generated before and after the campaign. The report include three key field like campaign name, total revenue before promotion, total revenue after promotion. This report should help in evaluating the financial impact of our promotional campaigns. Display the value in millions.

```
USE retail_events_db;

CREATE TEMPORARY TABLE temp_revenue_before_promo_final AS

SELECT

   d.campaign_id,

   d.campaign_name,

   SUM(p.base_price * p.Sold_quantity_before_promo) / 1000000 AS total_revenue_before_promo_final

FROM

   dim_campaigns d

JOIN

   fact_events p ON d.campaign_id = p.campaign_id

GROUP BY

   d.campaign_id, d.campaign_name;

CREATE TEMPORARY TABLE temp_revenue_after_promo AS

SELECT

   d.campaign_id,

   d.campaign_name,

   SUM(p.base_price * p.Sold_quantity_after_promo) / 1000000 AS total_revenue_after_promo

FROM

   dim_campaigns d

JOIN

   fact_events p ON d.campaign_id = p.campaign_id

GROUP BY

   d.campaign_id, d.campaign_name;

SELECT

   d.campaign_id,

   d.campaign_name,
```

```
    COALESCE(rbp.total_revenue_before_promo, 0) AS total_revenue_before_promo,
    COALESCE(rap.total_revenue_after_promo, 0) AS total_revenue_after_promo
FROM
    dim_campaigns d
LEFT JOIN
    temp_revenue_before_promo rbp ON d.campaign_id = rbp.campaign_id
LEFT JOIN
    temp_revenue_after_promo rap ON d.campaign_id = rap.campaign_id;
```

Request_4 Produce a report that calculates the incremental sold quantity (ISU%) for each category during Diwali campaign. Additionally provide rankings for the categories based on their ISU%. The report will include three key fields: category, isu% and rank order. This information will assist in assessing the category wise success and impact of the Diwali campaign on incremental sales.

Note: ISU % is calculated as the percentage increased/decreased in quantity sold (after promo) compared to quantity sold before promo.

```
USE retail_events_db;
-- Calculate incremental sold quantity (ISU%) for each category during Diwali campaign
WITH Diwali_Campaign AS (
    SELECT
        pc.category,
        SUM(fe.sold_quantity_after_promo) AS total_quantity_after_promo,
        SUM(fe.sold_quantity_before_promo) AS total_quantity_before_promo
    FROM
        fact_events fe
    JOIN
        dim_products pc ON fe.product_code = pc.product_code
    WHERE
        fe.campaign_id = 'CAMP_DIW_01'
    GROUP BY
        pc.category
)
```

```
-- Calculate ISU%

SELECT

   category,

   CASE

      WHEN total_quantity_before_promo = 0 THEN NULL -- Avoid division by zero

      ELSE ((total_quantity_after_promo - total_quantity_before_promo) / total_quantity_before_promo) * 100

   END AS isu_percentage,

   RANK() OVER (ORDER BY ((total_quantity_after_promo - total_quantity_before_promo) /
total_quantity_before_promo) DESC) AS rank_order

FROM

   Diwali_Campaign

ORDER BY

   isu_percentage DESC; -- Order by ISU% in descending order to get the highest first
```

Request_5 Create a report featu ring the Top 5 products, ranked by Incremental Revenue Percentage (IR%), across all campaigns. The report will provide essential information including product name, category, and ir%. This analysis helps identify the most successful products in terms of incremental revenue across our campaigns, assisting in product optimization.

```
USE retail_events_db;
WITH ProductIncrementalRevenue AS (
  SELECT
    dp.product_name,
    dp.category,
    SUM(fe.base_price * (fe.sold_quantity_after_promo - fe.sold_quantity_before_promo)) AS
total_incremental_revenue,
    SUM(fe.base_price * fe.sold_quantity_before_promo) AS total_revenue_before_promo
  FROM
    fact_events fe
  JOIN
    dim_products dp ON fe.product_code = dp.product_code
  GROUP BY
    dp.product_name, dp.category
)
-- Calculate IR% and rankings
SELECT
  product_name,
  category,
  CASE
    WHEN total_revenue_before_promo = 0 THEN NULL
    ELSE (total_incremental_revenue / total_revenue_before_promo) * 100
  END AS ir_percentage
FROM
  ProductIncrementalRevenue
ORDER BY
  ir_percentage DESC
LIMIT 5;
```