```
-- Create database retail_events_db;
use retail_events_db;

-- Provide a list of products with a base price greater than 500 and that are
-- featured in promo type of 'BOGOF'. This information will hwlp us identify
-- high-value products that are currently being heavily discounted, which can
-- be useful for evaluating our pricing and promotion strategies.

Select distinct(product_name), base_price, promo_type
from fact_events Join dim_products Using (product_code)
where base_price > 500 And promo_type = 'BOGOF';

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

Select city, Count(store_id) as Store_count from dim_stores
Group By city Order By Store_count;

-- Generate a report that displays each campaign along with total revenue
-- generated before and after the campaign? The report includes three key fields:
-- campaign_name, total_revenue(before_promotion), total_revenue(after_promotion).
-- This report should help in evaluating financial impact of our promotional campaigns.
--  (Display the values in Millions.)

-- SELECT
--     dc.campaign_name,
--     ROUND(
```

```
--        SUM(fe.base_price * fe.quantity_sold(before_promo)) / 1000000, 2
--    ) AS total_revenue_before_promo,
--    ROUND(
--        SUM(
--            CASE
--                WHEN fe.promo_type = 'BOGOF' THEN fe.base_price * 0.5 * (fe.quantity_sold(after_promo) * 2)
--                WHEN fe.promo_type = '500 Cashback' THEN (fe.base_price - 500) * fe.quantity_sold(after_promo)
--                WHEN fe.promo_type = '50% OFF' THEN fe.base_price * 0.5 * fe.quantity_sold(after_promo)
--                WHEN fe.promo_type = '33% OFF' THEN fe.base_price * 0.67 * fe.quantity_sold(after_promo)
--                WHEN fe.promo_type = '25% OFF' THEN fe.base_price * 0.75 * fe.quantity_sold(after_promo)
--            END
--        ) / 1000000, 2
--    ) AS total_revenue_after_promo
-- FROM
--    fact_events fe
-- JOIN
--    dim_campaigns dc ON fe.campaign_id = dc.campaign_id
-- GROUP BY
--    dc.campaign_name;


 SELECT
    campaign_name,
    ROUND(SUM(base_price * `quantity_sold(before_promo)`) / 1000000,2) AS total_revenue_before_promotion,
    ROUND(SUM(CASE
```

```
        WHEN promo_type = 'BOGOF' THEN base_price * 0.5 * (`quantity_sold(after_promo)` * 2)
        WHEN promo_type = '500 Cashback' THEN (base_price - 500) * `quantity_sold(after_promo)`
        WHEN promo_type = '50% OFF' THEN base_price * 0.5 * `quantity_sold(after_promo)`
        WHEN promo_type = '33% OFF' THEN base_price * 0.67 * `quantity_sold(after_promo)`
        WHEN promo_type = '25% OFF' THEN base_price * 0.75 * `quantity_sold(after_promo)` END) /
1000000,2) AS total_revenue_after_promotion
    FROM
    fact_events
    JOIN
    dim_campaigns USING (campaign_id)
    GROUP BY campaign_name;

-- Produce a report that calculates Incremental Sold Units (ISU%) for each
-- category during the diwali campaign. Additionally provide rankings for the
-- categories based on their ISU%. The report will include three key fields:
-- catgory, 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 increas/decrease in quantity sold
-- (after promo) compared to quantity sold (before promo).

With  Diwali_campaign_sale as ( Select category ,
    Round(Sum((
        Case
        When promo_type = "BOGOF" Then `quantity_sold(after_promo)`*2
        Else `quantity_sold(after_promo)`
        End
     - `quantity_sold(before_promo)`) * 100)
     / Sum(`quantity_sold(before_promo)`),2) as `ISU%`
     From fact_events
```

```
    Join
    dim_products using(product_code)
    Join
    dim_campaigns using(campaign_id)
    Where campaign_name = "Diwali"
    Group by category)

    Select
    Category ,
    `ISU%`,
    row_number() Over(order by `ISU%` desc) as rank_order
    From Diwali_campaign_sale ;

-- Create a report featuring 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.

 SELECT
   product_name,
   category,
   ROUND((SUM(CASE
        WHEN promo_type = 'BOGOF' THEN base_price * 0.5 * (`quantity_sold(after_promo)` * 2)
        WHEN promo_type = '500 Cashback' THEN (base_price - 500) * `quantity_sold(after_promo)`
        WHEN promo_type = '50% OFF' THEN base_price * 0.5 * `quantity_sold(after_promo)`
        WHEN promo_type = '33% OFF' THEN base_price * 0.67 * `quantity_sold(after_promo)`
        WHEN promo_type = '25% OFF' THEN base_price * 0.75 * `quantity_sold(after_promo)`
        ELSE 0
```

```
        END)
        - SUM(base_price * `quantity_sold(before_promo)`)) / SUM(base_price *
`quantity_sold(before_promo)`) * 100,2) AS `IR%`
    FROM
    fact_events
    JOIN
    dim_products USING (product_code)
    GROUP BY product_name , category
    ORDER BY `IR%` DESC
    LIMIT 5;
```