Select * from Information_schema.tables;

#Analysis

-- We used this query to list all the tables across all databases in our MySQL server.

-- It gives us a complete view of table names, types (like base tables or views), storage engines, row counts, and other structural details.

-- This helped us understand what data exists and how it’s organized before diving deeper into any specific table.

select * from Information_schema.columns

Where Table_Name = 'dim_customers';

#Analysis

-- We ran this query to get all the column-level details for the dim_customers table.

-- It shows each column’s name, data type, position, whether it allows NULLs, and other metadata.

-- This helped us understand the structure of the table — what kind of data each column holds and how it's defined in the database.

Select distinct Country from dim_customers;

#Analysis

-- We used this query to extract all the unique countries present in the dim_customers table.

-- It helped us quickly check which countries our customers belong to. We also noticed a value like 'n/a',

-- which could indicate missing or invalid data — something we might want to clean or investigate further.

Select distinct category, subcategory, product_name from dim_products

Order by 1,2,3;

#Analysis

-- We pulled all unique combinations of category, subcategory, and product names to understand the product hierarchy and avoid duplicates.

-- Ordered it for easier viewing.

select

min(order_date) as first_order_date,

max(order_date) as last_order_date,

YEAR(MAX(order_date)) - YEAR(MIN(order_date)) AS order_range_years

from fact_sales;

#Analysis

-- We checked the earliest and latest order dates in fact_sales to understand the time span of our data.

-- The dataset covers a 4-year sales period.

Select

min(birthdate) as oldest_birthdate,

YEAR(CURDATE()) - YEAR(MIN(birthdate)) AS oldest_age,

max(birthdate) as youngest_birthdate,

YEAR(CURDATE()) - YEAR(max(birthdate)) AS youngest_age

from dim_customers;

#Analysis

-- We checked the oldest and youngest birthdates in dim_customers to estimate customer age range.

-- The oldest customer is around 109 years old, and the youngest is about 39.

-- TOTAL SALES

select sum(sales_amount) as total_sales from fact_sales;

Analysis

-- We calculated the total sales from the fact_sales table.

-- The overall sales amount is $29,356,250.

-- Finding how many items are sold

select sum(quantity) as total_quantity from fact_sales;

Analysis

-- We summed up the total quantity of items sold from the fact_sales table.

-- A total of 60,423 units were sold.

-- Finding the average selling price

select avg(price) as avg_price from fact_sales;

Analysis

-- We calculated the average selling price per unit from the fact_sales table.

-- The average price is around $486.04.

-- Finding the total number of orders

select count(order_number) as total_orders from fact_sales;

select count(distinct(order_number)) as total_orders from fact_sales;

Analysis

-- We counted the total number of unique orders in the fact_sales table.

-- There are 27,659 distinct orders in the dataset.

-- Finding the total number of products

select count(product_key) as total_products from dim_products;

select count(distinct(product_key)) as total_products from dim_products;

Analysis

-- We counted the unique product entries in the dim_products table.

-- There are 295 distinct products in the dataset.

-- Finding total number of customers

select count(customer_key) as total_customers from dim_customers;

select count(distinct(customer_key)) as total_customers from dim_customers;

Analysis

-- We counted the number of unique customers in the dim_customers table.

-- The dataset contains 18,484 distinct customers.

-- Finding the total number of number of customers that has placed an order

select count(distinct(customer_key)) as total_customers from fact_sales;

Analysis

-- We counted how many unique customers have placed at least one order.

-- A total of 18,484 customers made purchases.

-- report showing all metrics in one

select 'Total Sales' as measure_name, sum(sales_amount) as measure_value from fact_sales

Union All

select 'Total Quantity' as measure_name, sum(quantity) as measure_value from fact_sales

Union All

select 'Avg Price' as measure_name, avg(price) as measure_value from fact_sales

Union All

select 'Total Orders' as measure_name, count(distinct(order_number)) as measure_value from fact_sales

Union All

select 'Total Products' as measure_name, count(distinct(product_key)) as measure_value from dim_products

Union All

select 'Total Customers' as measure_name, count(distinct(customer_key)) as measure_value from dim_customers;

-- Total number of customers per country

```
select country,
count(customer_key) as total_customers
from dim_customers
group by country
order by total_customers desc;
```

Analysis

-- We grouped customers by country to see where most of them are from.

-- The United States has the highest number of customers, followed by Australia and the UK.

-- Total customers by gender

```
select gender,
count(customer_key) as total_customers
from dim_customers
group by gender
order by total_customers desc;
```

Analysis

-- We counted customers by gender to understand the distribution.

-- The data is nearly balanced between male and female customers, with a few entries marked as 'n/a'.

-- total products by category

```
select category,
count(product_key) as total_products
from dim_products
group by category
order by total_products desc;
```

Analysis

-- We grouped products by category to see how many items fall under each.

-- 'Components' and 'Bikes' have the most products, while 7 entries have no category assigned.

-- avg cost in each category

select category,

avg(cost) as avg_cost

from dim_products

group by category

order by avg_cost desc;

Analysis

-- We calculated the average cost per product category.

-- Bikes are the most expensive on average, while Accessories and Clothing have the lowest average cost.

-- total revevenue for each category

select p.category,

sum(f.sales_amount) as total_revenue

from fact_sales f

left join dim_products p

on p.product_key = f.product_key

Group by p.category

Order by total_revenue desc;

Analysis

-- We calculated total revenue generated by each product category.

-- Bikes brought in the highest revenue by far, followed by Accessories and Clothing.

-- total revenue generated by each customer

```
select
c.customer_key,
c.first_name,
c.last_name,
sum(f.sales_amount) as total_revenue
from fact_sales f
left join dim_customers c
on c.customer_key = f.customer_key
group by
c.customer_key,
c.first_name,
c.last_name
Order by total_revenue desc;
```

Analysis

-- We calculated total revenue generated by each customer.

-- The top customers each contributed over $13,000 in sales, showing a few high-value buyers.

-- distribution of sold items across countries

```
select
c.country,
sum(f.quantity) as total_sold_items
from fact_sales f
left join dim_customers c
on c.customer_key = f.customer_key
group by c.country
Order by total_sold_items desc;
```

Analysis

-- We analyzed how sold items are distributed across countries.

-- The U.S. leads in total items sold, followed by Australia and Canada.

#ranking analysis - rank[dimension] by measure

-- top 5 products generating revenue

```
select
p.product_name,
sum(f.sales_amount) as total_revenue
from fact_sales f
left join dim_products p
on p.product_key = f.product_key
group by p.product_name
order by total_revenue Desc
limit 5;
```

Analysis

-- We identified the top 5 products by total revenue.

-- All top performers are variants of the Mountain-200 bike, indicating it’s the highest-selling product line.

-- worst 5 perfoming products

```
select
p.product_name,
sum(f.sales_amount) as total_revenue
from fact_sales f
left join dim_products p
on p.product_key = f.product_key
group by p.product_name
order by total_revenue
limit 5;
```

Analysis

-- We found the 5 lowest revenue-generating products.

-- These include low-cost items like socks and bike maintenance tools, contributing minimal sales.

-- Find the top 10 customers who have generated the highest revenue

```
SELECT
   c.customer_key,
   c.first_name,
   c.last_name,
   SUM(f.sales_amount) AS total_revenue
FROM fact_sales f
LEFT JOIN dim_customers c
   ON c.customer_key = f.customer_key
GROUP BY
   c.customer_key, c.first_name, c.last_name
ORDER BY total_revenue DESC
limit 10;
```

Analysis

-- We listed the top 10 customers by total revenue contribution.

-- All of them generated over $12,900 each, highlighting a strong group of high-value customers.

-- The 3 customers with the fewest orders placed

```
SELECT
   c.customer_key,
   c.first_name,
   c.last_name,
   COUNT(DISTINCT order_number) AS total_orders
FROM fact_sales f
LEFT JOIN dim_customers c
   ON c.customer_key = f.customer_key
GROUP BY
   c.customer_key,
   c.first_name,
   c.last_name
ORDER BY total_orders
limit 3;
```

Analysis

-- We identified the 3 customers with the fewest orders, each placing only 1 order.

-- This helps spot low-engagement customers for potential reactivation strategies.

-- Analyse sales performance over time

```
SELECT
    YEAR(order_date) AS order_year,
    MONTH(order_date) AS order_month,
    SUM(sales_amount) AS total_sales,
    COUNT(DISTINCT customer_key) AS total_customers,
    SUM(quantity) AS total_quantity
FROM fact_sales
WHERE order_date IS NOT NULL
GROUP BY YEAR(order_date), MONTH(order_date)
ORDER BY YEAR(order_date), MONTH(order_date);
```

Analysis

-- We analyzed monthly sales trends over the years.

-- This shows how total sales, unique customers, and quantity sold varied month by month, helping us spot seasonality or growth patterns.

-- SegmentING products into cost ranges and

-- countING how many products fall into each segment

```
WITH product_segments AS (
   SELECT
      product_key,
      product_name,
      cost,
      CASE
         WHEN cost < 100 THEN 'Below 100'
         WHEN cost BETWEEN 100 AND 500 THEN '100-500'
         WHEN cost BETWEEN 500 AND 1000 THEN '500-1000'
         ELSE 'Above 1000'
      END AS cost_range
   FROM dim_products
)
SELECT
   cost_range,
   COUNT(product_key) AS total_products
FROM product_segments
GROUP BY cost_range
ORDER BY total_products DESC;
```

Analysis

-- We segmented products into cost ranges to understand pricing distribution.

-- Most products fall under the ₹100–₹500 and below ₹100 categories, with fewer in higher price brackets.

-- Which categories contribute the most to overall sales?

```
WITH category_sales AS (
    SELECT
        p.category,
        SUM(f.sales_amount) AS total_sales
    FROM fact_sales f
    LEFT JOIN dim_products p
        ON p.product_key = f.product_key
    GROUP BY p.category
)
SELECT
    category,
    total_sales,
    SUM(total_sales) OVER () AS overall_sales,
    ROUND((CAST(total_sales AS FLOAT) / SUM(total_sales) OVER ()) * 100, 2) AS
percentage_of_total
FROM category_sales
ORDER BY total_sales DESC;
```

Analysis

-- We calculated how much each category contributes to total sales.

-- Bikes dominate with over 96% of all revenue, making them the key driver of business.