E-Commerce Sales Data

Basic Level Questions:

1.Write a query to display all products and their brands from the Train dataset.

- SELECT DISTINCT Product, Product_Brand
  FROM Train
  ORDER BY Product_Brand;

2. How many unique product brands are there in the dataset?

- SELECT COUNT(DISTINCT Product_Brand) as unique_brands
  FROM Train;

3. Calculate the average Item_Rating for all products.

- SELECT ROUND(AVG(Item_Rating), 2) as average_rating
  FROM Train;

4. List all products with an Item_Rating greater than 4.0

- SELECT Product, Product_Brand, Item_Rating
  FROM Train
  WHERE Item_Rating > 4.0
  ORDER BY Item_Rating DESC;

5. Find the number of products in each Item_Category.

- SELECT Item_Category, COUNT(*) as product_count
  FROM Train
  GROUP BY Item_Category
  ORDER BY product_count DESC;

Intermediate Level Questions:

6. Calculate the average selling price for each Product_Brand, ordered by average price in descending order.

- SELECT Product_Brand,
  ROUND(AVG(Selling_Price), 2) as avg_price,

```
COUNT(*) as product_count
FROM Train
GROUP BY Product_Brand
ORDER BY avg_price DESC;
```

7. Find the top 5 most expensive products along with their brands and categories.

- ```
  SELECT Product,
  Product_Brand,
  Item_Category,
  Selling_Price
  FROM Train
  ORDER BY Selling_Price DESC
  LIMIT 5;
  ```

8. Create a query to show the number of products launched (based on Date) in each month of the year.

- ```
  SELECT MONTH(Date) as month,
  COUNT(*) as product_count
  FROM Train
  GROUP BY MONTH(Date)
  ORDER BY month;
  ```

9. Find products that have an Item_Rating above the average rating.

- ```
  WITH avg_rating AS (
    SELECT AVG(Item_Rating) as avg_item_rating
    FROM Train
  )
  SELECT t.Product,
      t.Product_Brand,
      t.Item_Rating
  FROM Train t, avg_rating ar
  WHERE t.Item_Rating > ar.avg_item_rating
  ORDER BY t.Item_Rating DESC;
  ```

10. For each Subcategory_1, calculate the minimum, maximum, and average selling price.

- ```
  SELECT Subcategory_1,
  ```

```
MIN(Selling_Price) as min_price,
MAX(Selling_Price) as max_price,
ROUND(AVG(Selling_Price), 2) as avg_price
FROM Train
GROUP BY Subcategory_1
ORDER BY avg_price DESC;
```

Advanced Level Questions:

11. Create a query to find the Product_Brand that has products across the maximum number of different Item_Categories.

- ```
  SELECT Product_Brand,
  COUNT(DISTINCT Item_Category) as category_count
  FROM Train
  GROUP BY Product_Brand
  ORDER BY category_count DESC
  LIMIT 1;
  ```

12. Calculate the price difference between each product's selling price and the average selling price in its Item_Category (use window functions).

- ```
  WITH category_avg AS (
    SELECT Item_Category,
        AVG(Selling_Price) as avg_category_price
    FROM Train
    GROUP BY Item_Category
  )
  SELECT t.Product,
      t.Item_Category,
      t.Selling_Price,
      ROUND(ca.avg_category_price, 2) as category_avg_price,
      ROUND(t.Selling_Price - ca.avg_category_price, 2) as price_difference
  FROM Train t
  JOIN category_avg ca ON t.Item_Category = ca.Item_Category
  ORDER BY price_difference DESC;
  ```

13. Find products where the selling price is more than twice the average price in their respective Subcategory_2.

- ```
  WITH subcategory_avg AS (
    SELECT Subcategory_2,
        AVG(Selling_Price) as avg_price
    FROM Train
  ```
  ```

```
    GROUP BY Subcategory_2
)
SELECT t.Product,
    t.Subcategory_2,
    t.Selling_Price,
    ROUND(sa.avg_price, 2) as avg_subcategory_price
FROM Train t
JOIN subcategory_avg sa ON t.Subcategory_2 = sa.Subcategory_2
WHERE t.Selling_Price > 2 * sa.avg_price
ORDER BY t.Selling_Price DESC;
```

14. Create a query to rank products within each Item_Category based on their selling price (use ROW_NUMBER() or RANK()).

- ```
  SELECT Product,
   Item_Category,
  Selling_Price,
  RANK() OVER (PARTITION BY Item_Category ORDER BY Selling_Price DESC) as
  price_rank
  FROM Train;
  ```

15. Find the moving average of selling prices over the last 3 products (ordered by date) within each Product_Brand.

- ```
  SELECT Product_Brand,
      Product,
      Selling_Price,
      ROUND(AVG(Selling_Price) OVER (
        PARTITION BY Product_Brand
        ORDER BY Date
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
      ), 2) as moving_avg_price
  FROM Train
  ORDER BY Product_Brand, Date;
  ```