# Retail Orders Analysis

# Introduction

- The retail orders analysis focuses on sales performance across various time periods, regions and product categories, etc.
- By examining order volume fluctuations, temporal customer demand patterns can be obtained.
- The analysis aims to optimize strategies to enhance inventory planning and maximize revenue generation during critical timeframes.

# Analysis Steps

1. Define Objectives
2. Identify Data Source
3. Data Collection and Cleaning
4. Data Processing
5. Draw insights
6. Test and Validate

# Objective

The retail orders analysis attempts to obtains insights from the data in line with answering the following:

1. Top 10 highest revenue generating products.
2. Top 5 highest selling products in each region.
3. Month over month growth comparison in sales for the year 2022 and 2023.
4. Month with the highest sales in each category.
5. Sub-category with the highest growth in terms of profit in 2023 in comparison to 2022.

# Dataset and Technology

- Dataset:
  - File Type : .csv
  - Source: https://www.kaggle.com/datasets/ankitbansal06/retail-orders
  - 9994 rows
  - Number of fields - 17
  - Field list: order_id, order_date, ship_mode, country, category, sub_category, product_id, list_price, quantity and discount_percent etc.
- Technology:
  - Python, MySQL

# Python

- Libraries and modules used:
  - pandas
  - zipfile
- Functions used:
  - read_csv()
  - head()
  - to_datetime()
  - drop()
  - to_csv()

# MySQL Queries

1. FIND TOP 10 HIGHEST REVENUE GENERATING PRODUCTS

```
select
        product_id,
        sum(final_sale) as sales
from orders
group by product_id
order by sum(final_sale) desc
limit 10;
```

# MySQL Queries

2. FIND TOP 5 HIGHEST SELLING PRODUCTS IN EACH REGION

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

# MySQL Queries

3. FIND MONTH OVER MONTH GROWTH COMPARISON FOR 2022 AND 2023

```
with cte as
(       select
                left(order_date,4) as year,
                substring(order_date,6,2) as month,
                sum(sale_price) as sales
        from orders
        group by left(order_date,4), substring(order_date,6,2)
)
select *
from cte c1, cte c2
where c1.month = c2.month and c1.year != c2.year and c1.year > c2.year;
```

# MySQL Queries

4. DETERMINE THE MONTH WITH HIGHEST SALES IN EACH CATEGORY

```
with cte as
(select
        category,
        substring(order_date,1,4) as year,
        substring(order_date,6,2) as month,
        sum(sale_price) as sales,
        row_number() over(partition by category order by sum(sale_price) desc) as rn
from orders
group by category, substring(order_date,1,4) , substring(order_date,6,2))
select *
from cte
where rn = 1;
```

# MySQL Queries

5. WHICH SUB-CATEGORY HAD THE HIGHEST GROWTH BY PROFIT

IN 2023 COMPARED TO 2022

```
with cte as
(       select sub_category,
                left(order_date,4) as year,
                 sum(profit) as sales
        from orders
        group by sub_category, left(order_date,4))
select *, (c1.sales - c2.sales)*100/c1.sales as growth_percent
from cte c1, cte c2
where c1.sub_category = c2.sub_category and c1.year > c2.year
order by growth_percent desc
limit 1;
```