

Unit economics

ANALYZING BUSINESS DATA IN SQL



Michel Semaan
Data Scientist

Unit economics

- **Unit economics:** Measures performance per unit, as opposed to overall performance
- **Example:** Average Revenue Per User (ARPU)
 - **Formula:** $\frac{Revenue}{Count\ of\ users}$
 - **Use:** Measures a company's success in scaling its business model

ARPU - query (I)

```
WITH kpis AS (  
  SELECT  
    SUM(meal_price * order_quantity) AS revenue,  
    COUNT(DISTINCT user_id) AS users  
  FROM meals  
  JOIN orders ON meals.meal_id = orders.meal_id  
  
  SELECT  
    ROUND(  
      revenue :: NUMERIC / GREATEST(users, 1),  
      2) AS arpu  
  FROM kpis;
```

ARPU - query (I) by month

```
WITH kpis AS (  
  SELECT  
    DATE_TRUNC('month', order_date) AS delivr_month,  
    SUM(meal_price * order_quantity) AS revenue,  
    COUNT(DISTINCT user_id) AS users  
  FROM meals  
  JOIN orders ON m.meal_id = o.meal_id  
  GROUP BY delivr_month)  
  
SELECT  
  delivr_month,  
  ROUND(  
    revenue :: NUMERIC / GREATEST(users, 1),  
    2) AS arpu  
FROM kpis  
ORDER BY delivr_month ASC;
```

ARPU - query (II)

```
WITH user_revenues AS (  
  SELECT  
    user_id,  
    SUM(meal_price * order_quantity) AS revenue  
  FROM meals  
  JOIN orders ON meals.meal_id = orders.meal_id  
  GROUP BY user_id)  
  
SELECT  
  ROUND(AVG(revenue) :: NUMERIC, 2) AS arpu  
FROM user_revenues;
```

Comparing the two ways

First way

revenue	users
-----	-----
260226.75	1304

Second way

user_id	revenue
-----	-----
0	262.75
1	160.5
2	255.25

ARPU - result

```
arpu  
-----  
199.56
```

Unit economics

ANALYZING BUSINESS DATA IN SQL

Histograms

ANALYZING BUSINESS DATA IN SQL



Michel Semaan
Data Scientist

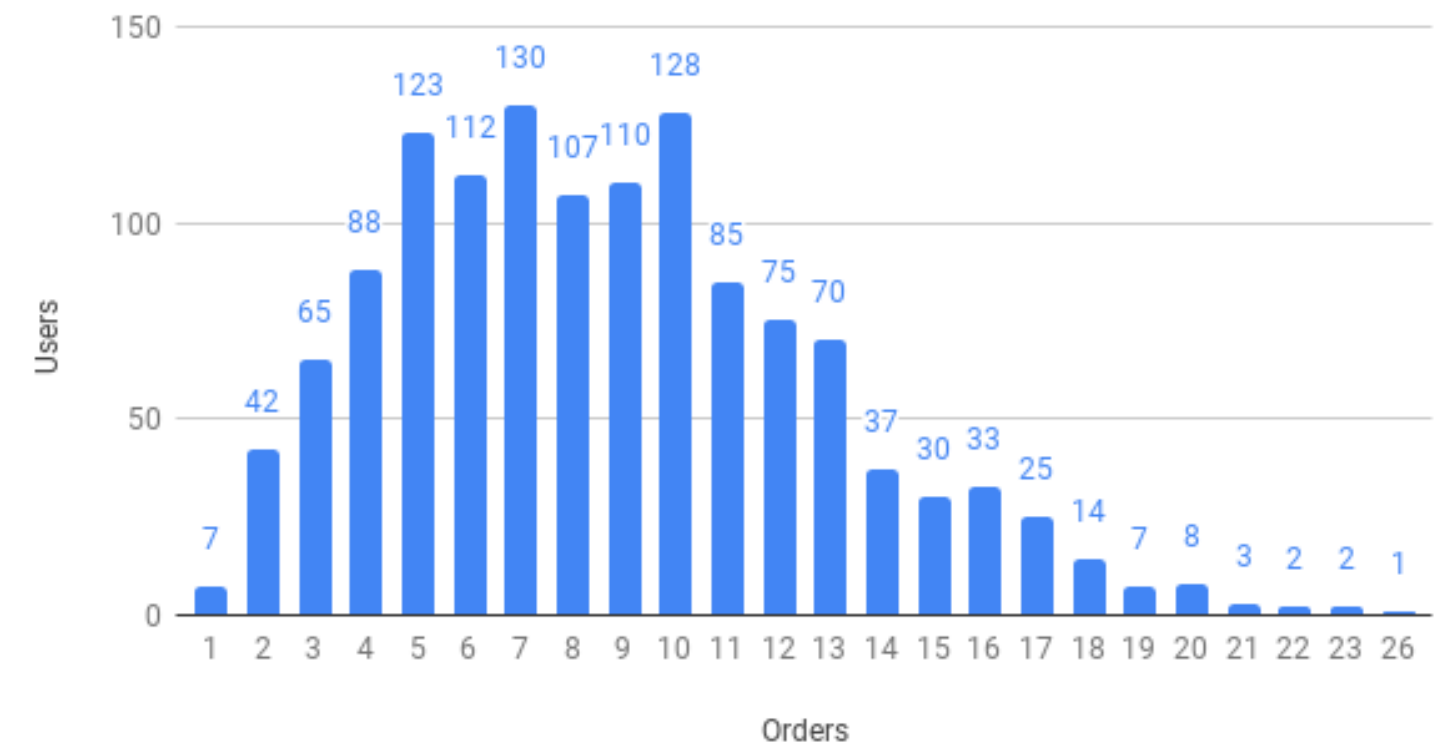
Histograms - overview

Histogram: Visualizes the frequencies of each value in a dataset

Frequency table

orders	users
1	7
2	42
3	65
...	...

Histogram of orders per user



Histograms - query (I)

```
WITH user_orders AS (  
  SELECT  
    user_id,  
    COUNT(DISTINCT order_id) AS orders  
  FROM meals  
  JOIN orders ON meals.meal_id = orders.meal_id  
  GROUP BY user_id)  
  
SELECT  
  orders,  
  COUNT(DISTINCT user_id) AS users  
FROM user_orders  
GROUP BY orders  
ORDER BY orders ASC;
```

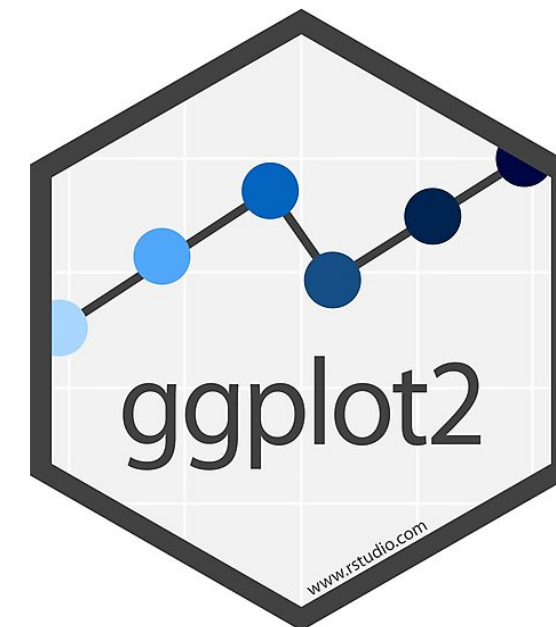
Histograms - query (II)

```
WITH user_revenues AS (  
  SELECT  
    user_id,  
    SUM(meal_price * order_quantity) AS revenue  
  FROM meals  
  JOIN orders ON meals.meal_id = orders.meal_id  
  GROUP BY user_id)  
  
SELECT  
  ROUND(revenue :: NUMERIC, -2) AS revenue_100,  
  COUNT(DISTINCT user_id) AS users  
FROM user_revenues  
GROUP BY revenue_100  
ORDER BY revenue_100 ASC;
```

Plotting histograms

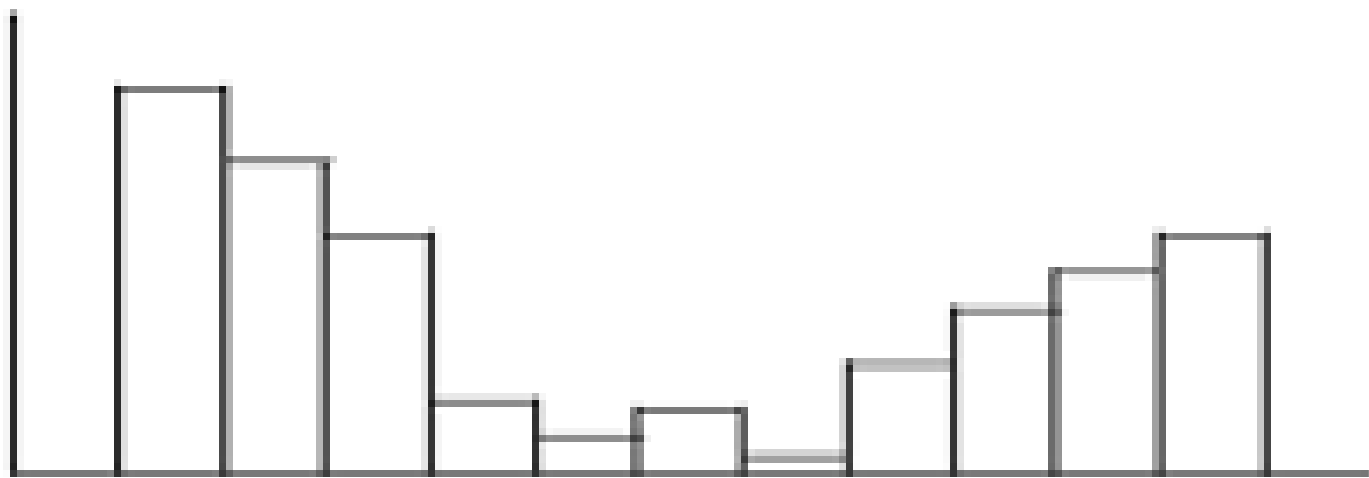


matplotlib



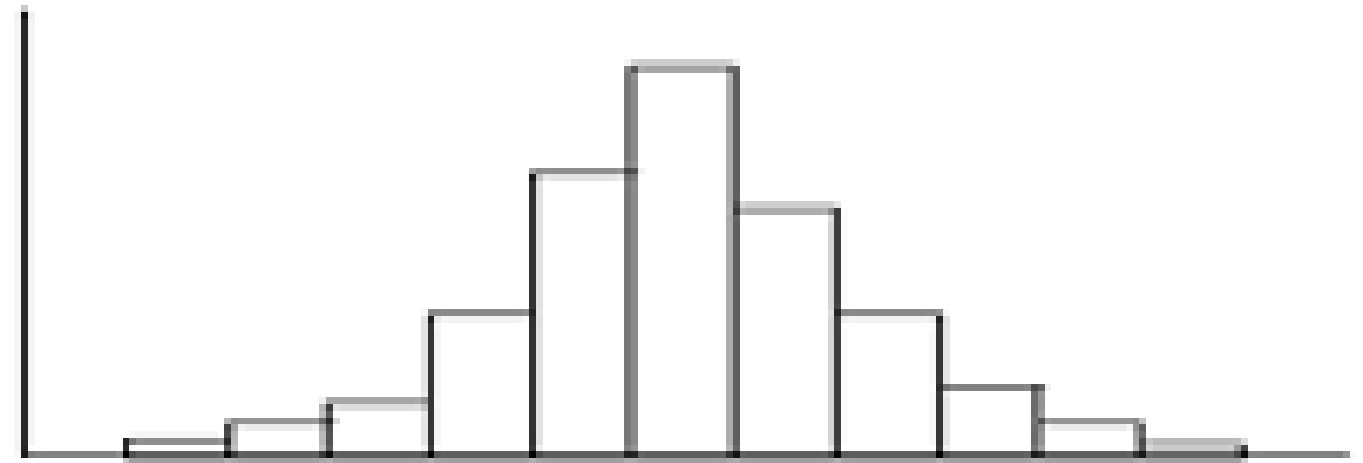
What do histograms tell you?

U-shaped histogram



- Many users who generate low and high levels of revenue; few who generate median level of revenue

Normal histogram



- Many users who generate median level of revenue; few who generate low or high levels of revenue

Histograms

ANALYZING BUSINESS DATA IN SQL

Bucketing

ANALYZING BUSINESS DATA IN SQL



Michel Semaan
Data Scientist

Histograms - recap

```
WITH user_orders AS (  
  SELECT  
    user_id,  
    COUNT(DISTINCT order_id) AS orders  
  FROM meals  
  JOIN orders ON meals.meal_id = orders.meal_id  
  GROUP BY user_id)  
  
SELECT  
  orders,  
  COUNT(DISTINCT user_id) AS users  
FROM user_orders  
GROUP BY orders  
ORDER BY orders ASC  
LIMIT 5;
```

CASE expression

Query

```
SELECT
  CASE
    WHEN meal_price < 4 THEN 'Low-price meal'
    WHEN meal_price < 6 THEN 'Mid-price meal'
    ELSE 'High-price meal'
  END AS price_category,
  COUNT(DISTINCT meal_id)
FROM meals
GROUP BY price_category;
```

Bucketing - query

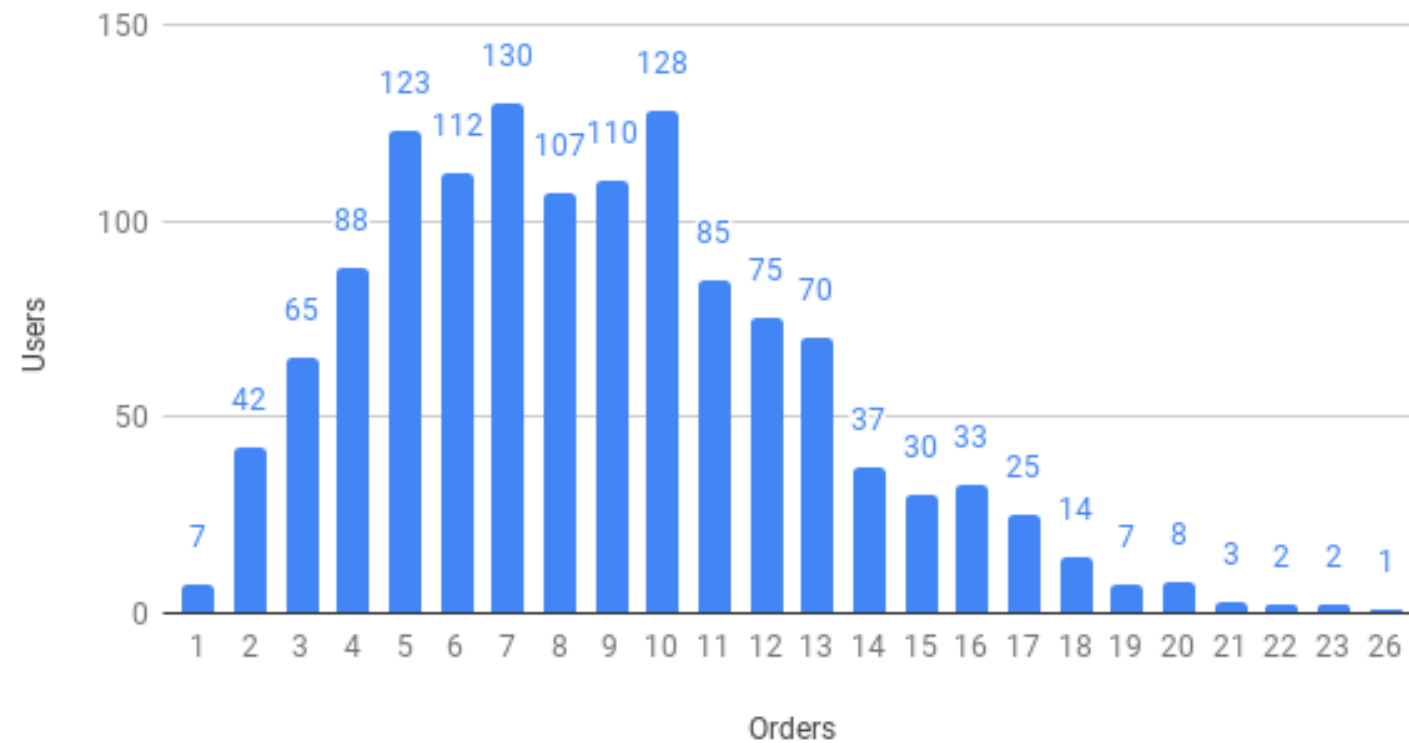
```
WITH user_revenues AS (  
  SELECT  
    user_id,  
    SUM(meal_price * order_quantity) AS revenue  
  FROM meals  
  JOIN orders ON meals.meal_id = orders.meal_id  
  GROUP BY user_id)  
  
SELECT  
  CASE  
    WHEN revenue < 150 THEN 'Low-revenue users'  
    WHEN revenue < 300 THEN 'Mid-revenue users'  
    ELSE 'High-revenue users'  
  END AS revenue_group,  
  COUNT(DISTINCT user_id) AS users  
FROM user_revenues  
GROUP BY revenue_group;
```

Bucketing - result

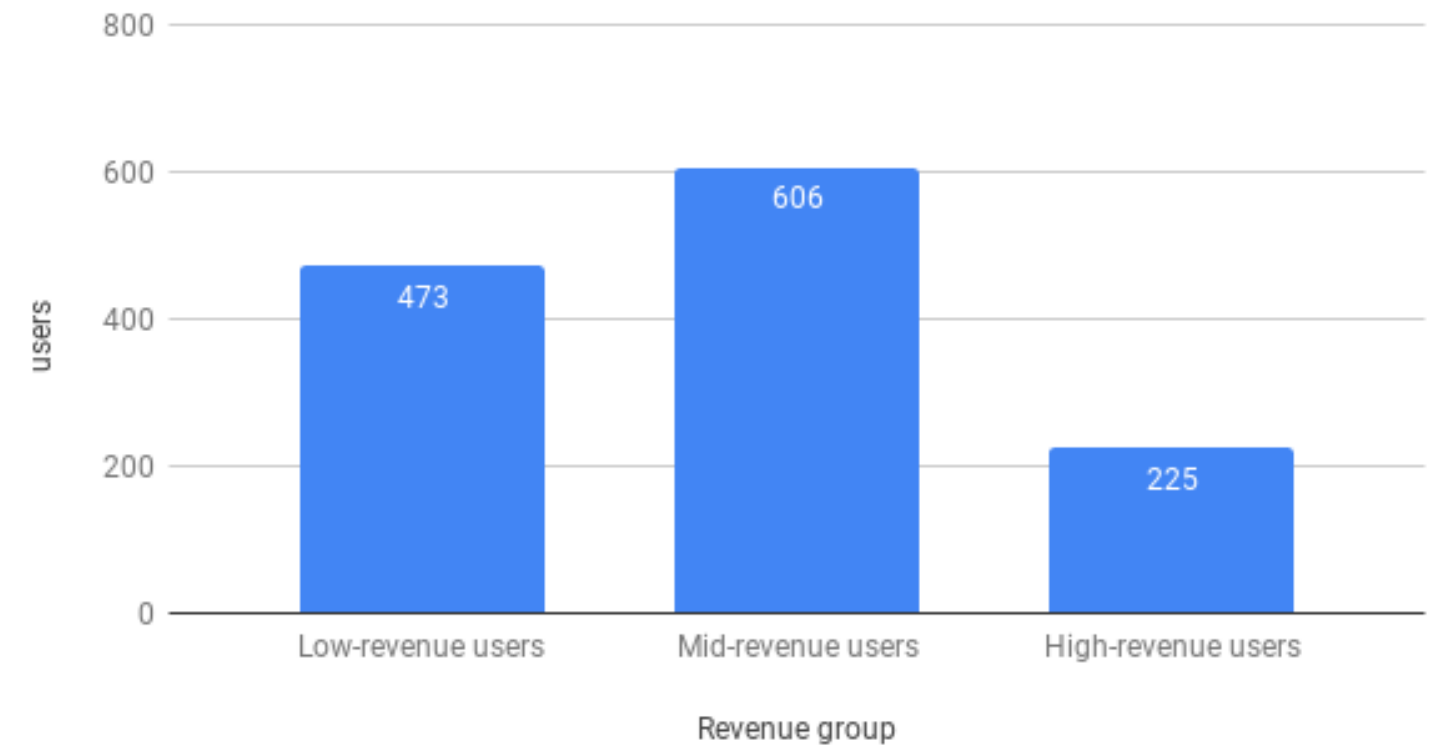
revenue_group	users
Low-revenue users	473
Mid-revenue users	606
High-revenue users	225

Histograms versus bar graphs

Histogram of orders per user



Users per revenue group



Bucketing

ANALYZING BUSINESS DATA IN SQL

Percentiles

ANALYZING BUSINESS DATA IN SQL



Michel Semaan
Data Scientist

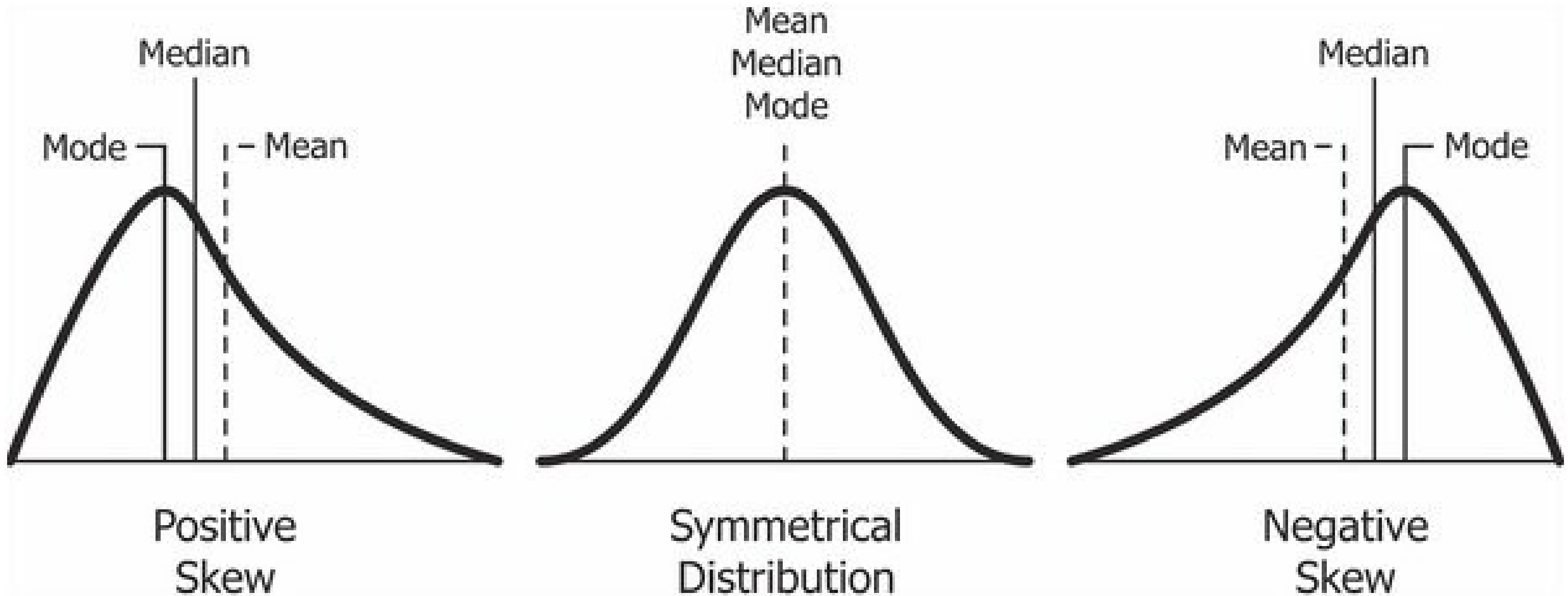
Percentiles - overview

- **Percentile:** n th percentile is the value for which $n\%$ of a dataset's values are beneath this value
 - Lowest value is the 0th percentile
 - Highest value is the 99th percentile

Quartiles

- **Example:** 25th percentile of user orders is 17, then 25% have ordered 17 times or less
- **First quartile:** 25th percentile
- **Third quartile:** 75th percentile
- **Interquartile range (IQR):** All data between the first and third quartiles
- **Second quartile:** 50th percentile, median
 - Different from the mean!

Skewed data



Quartiles - query

```
WITH user_orders AS (  
  SELECT  
    user_id,  
    COUNT(DISTINCT order_id) AS orders  
  FROM orders  
  GROUP BY user_id)  
  
SELECT  
  ROUND(  
    PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY orders ASC) :: NUMERIC,  
    2) AS orders_p25,  
  ROUND(  
    PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY orders ASC) :: NUMERIC,  
    2) AS orders_p50,  
  ROUND(  
    PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY orders ASC) :: NUMERIC,  
    2) AS orders_p75,  
  ROUND(AVG(orders) :: NUMERIC, 2) AS avg_orders  
FROM user_orders;
```

Quartiles - result

orders_p25	orders_p50	orders_p75	avg_orders
6.00	8.00	11.00	8.70

Percentiles

ANALYZING BUSINESS DATA IN SQL