SQL Case Study: VIP Customers & Revenue Concentration (SQLite + pandas)

by Konrad Kuleta

Scenario: Synthetic e-commerce dataset (CSV extracts) loaded into a local SQLite database for analysis.

Objective: Demonstrating advanced SQL (CTEs + window functions) by answering business-style questions:

- How many customers generate **~80% of total revenue**?
- What revenue share is generated by the **Top 1%**, **Top 5%**, and **Top 10%** customers?
- Who are the **VIP customers**, and what cumulative revenue share do they represent?

Workflow (high level):

1. Loading raw CSV files with pandas.
2. Cleaning numeric fields (EUR to float).
3. Loading the data into SQLite tables.
4. Running SQL queries and inspecting results via pandas.

In [1]:

```
# Data import
import sqlite3
import pandas as pd

con = sqlite3.connect("ecommerce_customers.db")

orders = pd.read_csv("fact_orders_2023_2025.csv")
customers = pd.read_csv("dim_customers_2023_2025.csv")
spend = pd.read_csv("fact_marketing_spend_daily_2023_2025.csv")
```

In [2]:

```
# Data formatting
def eur_to_float(s):
    # "240,53 €" -> 240.53
    return (s.astype(str)
              .str.replace("€", "", regex=False)
              .str.replace(" ", "", regex=False)
              .str.replace(".", "", regex=False)   # thousands separator
              .str.replace(",", ".", regex=False)  # comma to dot
              .astype(float))

orders["order_net_revenue"] = eur_to_float(orders["order_net_revenue"])
```

In [ ]:

```
# SQL data load
orders.to_sql("fact_orders", con, index=False, if_exists="replace")
customers.to_sql("dim_customers", con, index=False, if_exists="replace")
spend.to_sql("fact_marketing_spend_daily", con, index=False, if_exists="replace")
```

80/20 revenue concentration (how many customers drive 80% of revenue?)

In [6]:

```
sql = """
WITH customer_rev AS (
  SELECT
    customer_id,
    COUNT(*) AS orders,
    SUM(order_net_revenue) AS revenue
  FROM fact_orders
  GROUP BY customer_id
),
ranked AS (
  SELECT
    customer_id,
    orders,
    revenue,
    ROW_NUMBER() OVER (ORDER BY revenue DESC) AS rn,
    COUNT(*) OVER () AS n_customers,
    SUM(revenue) OVER () AS total_revenue,
    SUM(revenue) OVER (
      ORDER BY revenue DESC
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS cum_revenue
  FROM customer_rev
)
```

```
SELECT
  MIN(rn) AS customers_needed_for_80pct_revenue,
  ROUND(100.0 * MIN(rn) / MAX(n_customers), 2) AS pct_of_customers
FROM ranked
WHERE 1.0 * cum_revenue / total_revenue >= 0.80;
"""
pd.read_sql_query(sql, con)
```

```
Out[6]:    customers_needed_for_80pct_revenue   pct_of_customers
        0                                8798              48.88
```

Revenue concentration: share of revenue from Top 1% / 5% / 10% customers

```
In [7]: §sql = """
WITH customer_rev AS (
  SELECT
    customer_id,
    SUM(order_net_revenue) AS revenue
  FROM fact_orders
  GROUP BY customer_id
),
ranked AS (
  SELECT
    customer_id,
    revenue,
    ROW_NUMBER() OVER (ORDER BY revenue DESC) AS rn,
    COUNT(*)    OVER () AS n_customers,
    SUM(revenue) OVER () AS total_revenue
  FROM customer_rev
)
SELECT
  ROUND(
    100.0 * SUM(CASE WHEN rn <= CAST(n_customers * 0.01 + 0.999999 AS INT) THEN revenue ELSE 0 END)
    / MAX(total_revenue), 2
  ) AS top_1pct_revenue_share,
  ROUND(
    100.0 * SUM(CASE WHEN rn <= CAST(n_customers * 0.05 + 0.999999 AS INT) THEN revenue ELSE 0 END)
    / MAX(total_revenue), 2
  ) AS top_5pct_revenue_share,
  ROUND(
    100.0 * SUM(CASE WHEN rn <= CAST(n_customers * 0.10 + 0.999999 AS INT) THEN revenue ELSE 0 END)
    / MAX(total_revenue), 2
  ) AS top_10pct_revenue_share
FROM ranked;
"""
pd.read_sql_query(sql, con)
```

Out[7]:

	top_1pct_revenue_share	top_5pct_revenue_share	top_10pct_revenue_share
0	4.5	17.15	29.12

VIP list: Top 20 customers with revenue share and cumulative revenue share

In [8]:

```
sql = """
WITH customer_rev AS (
  SELECT
    customer_id,
    COUNT(*) AS orders,
    SUM(order_net_revenue) AS revenue
  FROM fact_orders
  GROUP BY customer_id
),
ranked AS (
  SELECT
    customer_id,
    orders,
    revenue,
    ROW_NUMBER() OVER (ORDER BY revenue DESC) AS rn,
    SUM(revenue) OVER () AS total_revenue,
    SUM(revenue) OVER (
      ORDER BY revenue DESC
      ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS cum_revenue
  FROM customer_rev
)
SELECT
  rn AS vip_rank,
  customer_id,
  orders,
  ROUND(revenue, 2) AS revenue,
  ROUND(100.0 * revenue / total_revenue, 2) AS revenue_share_pct,
  ROUND(100.0 * cum_revenue / total_revenue, 2) AS cumulative_share_pct
FROM ranked
ORDER BY rn
LIMIT 20;
"""
pd.read_sql_query(sql, con)
```

Out[8]:

	vip_rank	customer_id	orders	revenue	revenue_share_pct	cumulative_share_pct
0	1	1987	11	2870.96	0.04	0.04
1	2	12915	18	2861.05	0.04	0.07
2	3	12942	15	2860.94	0.04	0.11
3	4	10102	12	2835.46	0.04	0.15
4	5	5858	10	2721.90	0.03	0.18
5	6	7531	17	2671.56	0.03	0.21
6	7	9392	10	2551.26	0.03	0.25
7	8	14551	8	2531.85	0.03	0.28
8	9	8193	12	2528.20	0.03	0.31
9	10	15302	13	2502.84	0.03	0.34
10	11	7785	11	2496.91	0.03	0.38
11	12	15918	11	2486.75	0.03	0.41
12	13	14645	10	2479.32	0.03	0.44
13	14	8837	14	2352.56	0.03	0.47
14	15	4988	15	2337.36	0.03	0.50
15	16	8005	11	2327.77	0.03	0.53
16	17	4824	12	2310.29	0.03	0.56
17	18	12700	8	2278.09	0.03	0.59
18	19	9906	13	2277.01	0.03	0.62
19	20	1072	8	2256.23	0.03	0.65

Findings (high level)

- Revenue tends to be **concentrated in a small customer segment** (Pareto-like distribution).
- The notebook quantifies concentration via **80% revenue threshold** and **Top 1/5/10% revenue shares**.
- A **VIP customer list** highlights the top revenue contributors and their cumulative impact.

Notes

- Results depend on the dataset in `data/`. If the data is regenerated, the exact numbers may change.