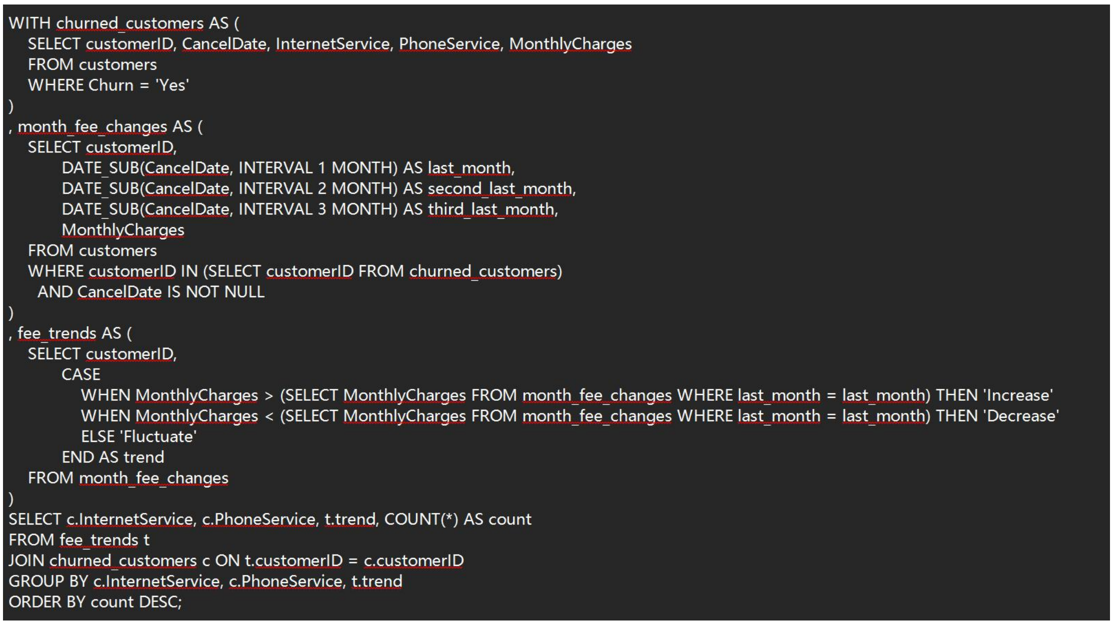
2（3）

我用的 chatgpt 4omini 模型

对于每个客户，计算他们在流失前 3 个月内的月费变化趋势(递增/递减/波动)，并与他们使用的主要服务(互联网/电话)做交叉分析，找出哪种服务组合的客户最容易因价格上涨而流失。″，chatgpt 给的代码如下：



```
WITH churned_customers AS (
   SELECT customerID, CancelDate, InternetService, PhoneService, MonthlyCharges
   FROM customers
   WHERE Churn = 'Yes'
)
, month_fee_changes AS (
   SELECT customerID,
          DATE_SUB(CancelDate, INTERVAL 1 MONTH) AS last_month,
          DATE_SUB(CancelDate, INTERVAL 2 MONTH) AS second_last_month,
          DATE_SUB(CancelDate, INTERVAL 3 MONTH) AS third_last_month,
          MonthlyCharges
   FROM customers
   WHERE customerID IN (SELECT customerID FROM churned_customers)
     AND CancelDate IS NOT NULL
)
, fee_trends AS (
   SELECT customerID,
          CASE
             WHEN MonthlyCharges > (SELECT MonthlyCharges FROM month_fee_changes WHERE last_month = last_month) THEN 'Increase'
             WHEN MonthlyCharges < (SELECT MonthlyCharges FROM month_fee_changes WHERE last_month = last_month) THEN 'Decrease'
             ELSE 'Fluctuate'
          END AS trend
   FROM month_fee_changes
)
SELECT c.InternetService, c.PhoneService, t.trend, COUNT(*) AS count
FROM fee_trends t
JOIN churned_customers c ON t.customerID = c.customerID
GROUP BY c.InternetService, c.PhoneService, t.trend
ORDER BY count DESC;
```

问题解析：

1.WHERE last_month = last_month 是永远成立的条件（相当于 WHERE 1=1），因为这个条件是在拿列自己和自己比较

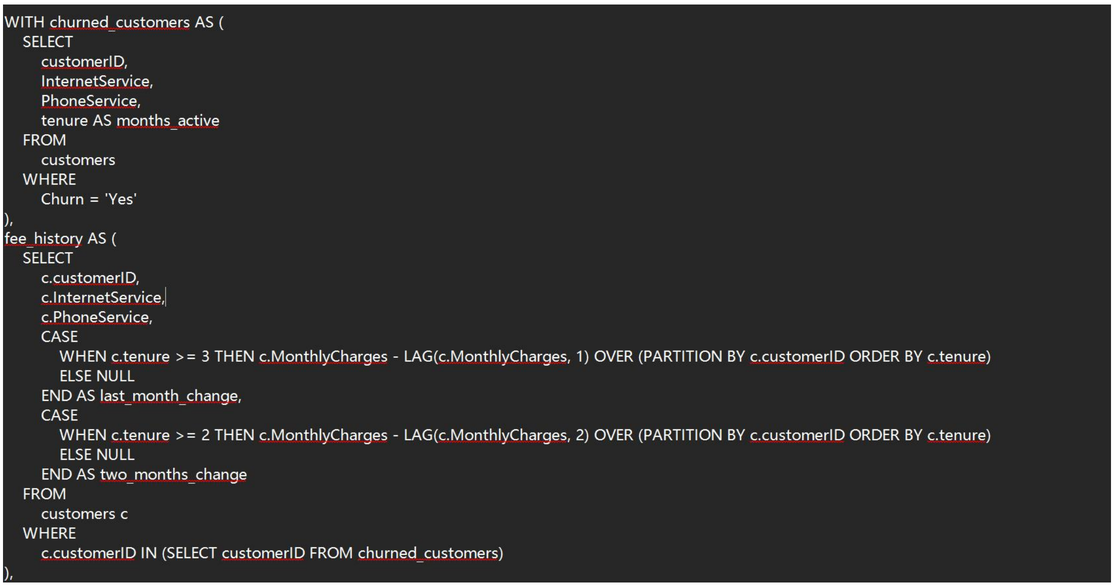
2. 子查询 (SELECT MonthlyCharges FROM month_fee_changes WHERE last_month = last_month) 会返回多行结果，但 CASE WHEN 需要

单值比较

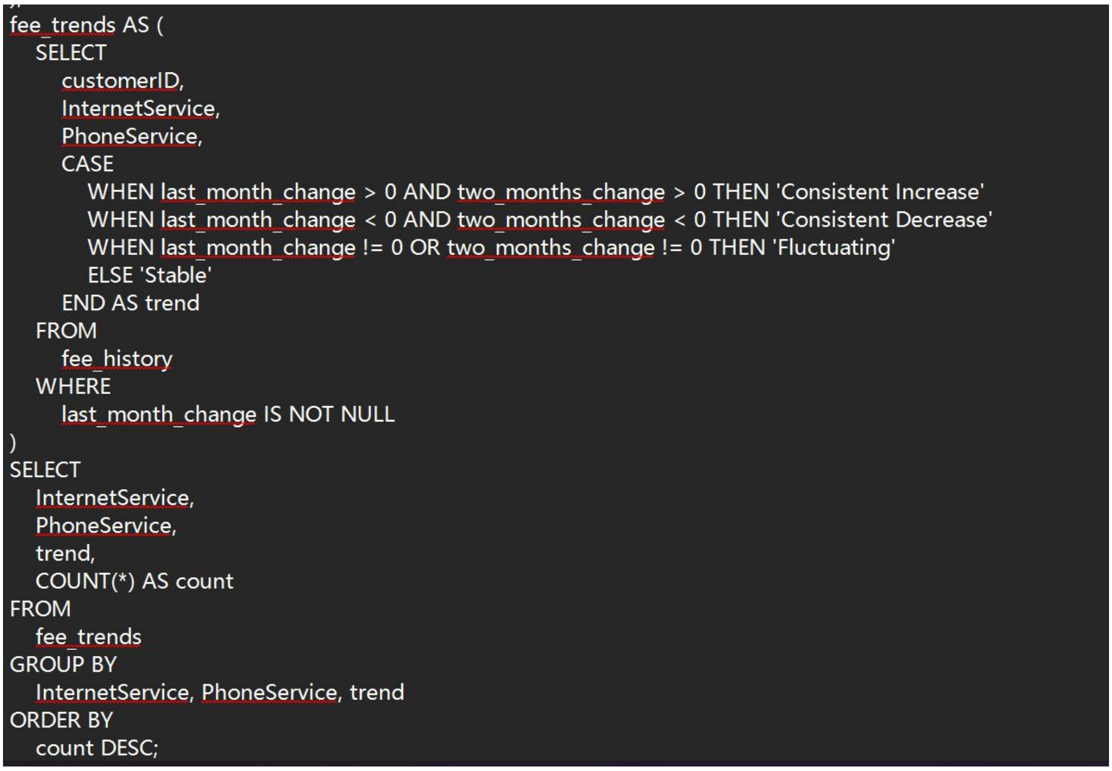
即使强制限制为单行，也无法获取"上个月"的费用，只是重复当前行数据

3. 没有时间序列数据：只有当前月费，没有历史月费记录，无法建立时间关联：不知道哪个月费对应哪个月

正确代码如下

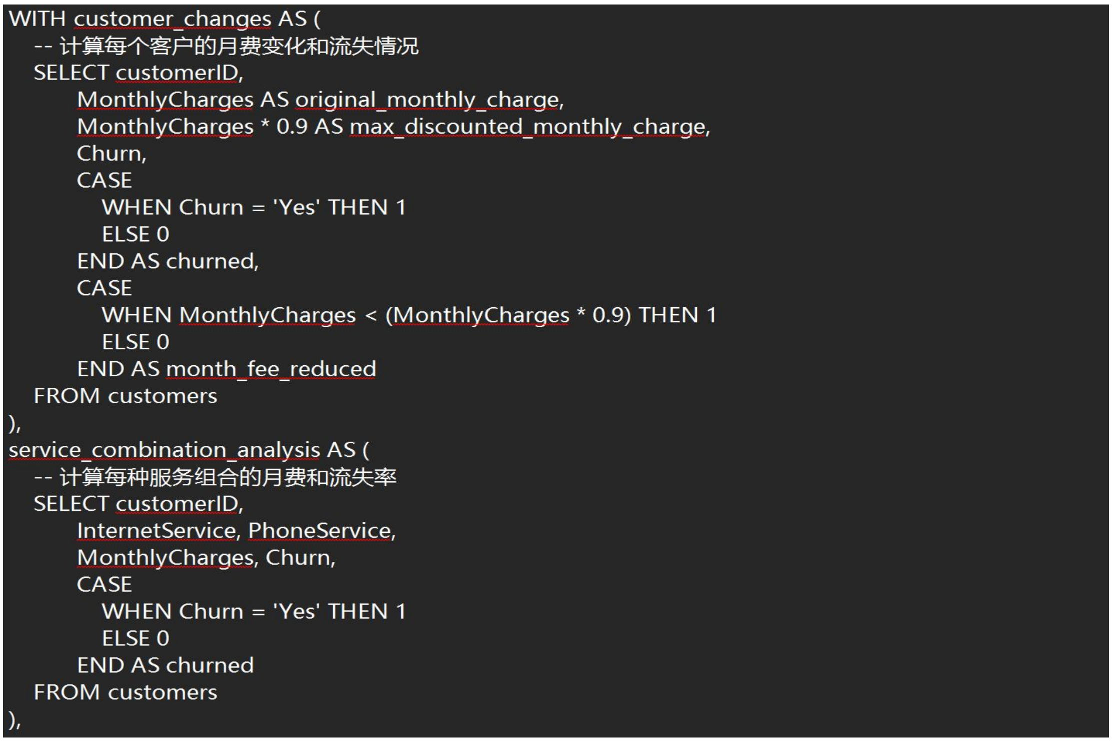


```sql
WITH churned_customers AS (
    SELECT
        customerID,
        InternetService,
        PhoneService,
        tenure AS months_active
    FROM
        customers
    WHERE
        Churn = 'Yes'
),
fee_history AS (
    SELECT
        c.customerID,
        c.InternetService,
        c.PhoneService,
        CASE
            WHEN c.tenure >= 3 THEN c.MonthlyCharges - LAG(c.MonthlyCharges, 1) OVER (PARTITION BY c.customerID ORDER BY c.tenure)
            ELSE NULL
        END AS last_month_change,
        CASE
            WHEN c.tenure >= 2 THEN c.MonthlyCharges - LAG(c.MonthlyCharges, 2) OVER (PARTITION BY c.customerID ORDER BY c.tenure)
            ELSE NULL
        END AS two_months_change
    FROM
        customers c
    WHERE
        c.customerID IN (SELECT customerID FROM churned_customers)
),
```

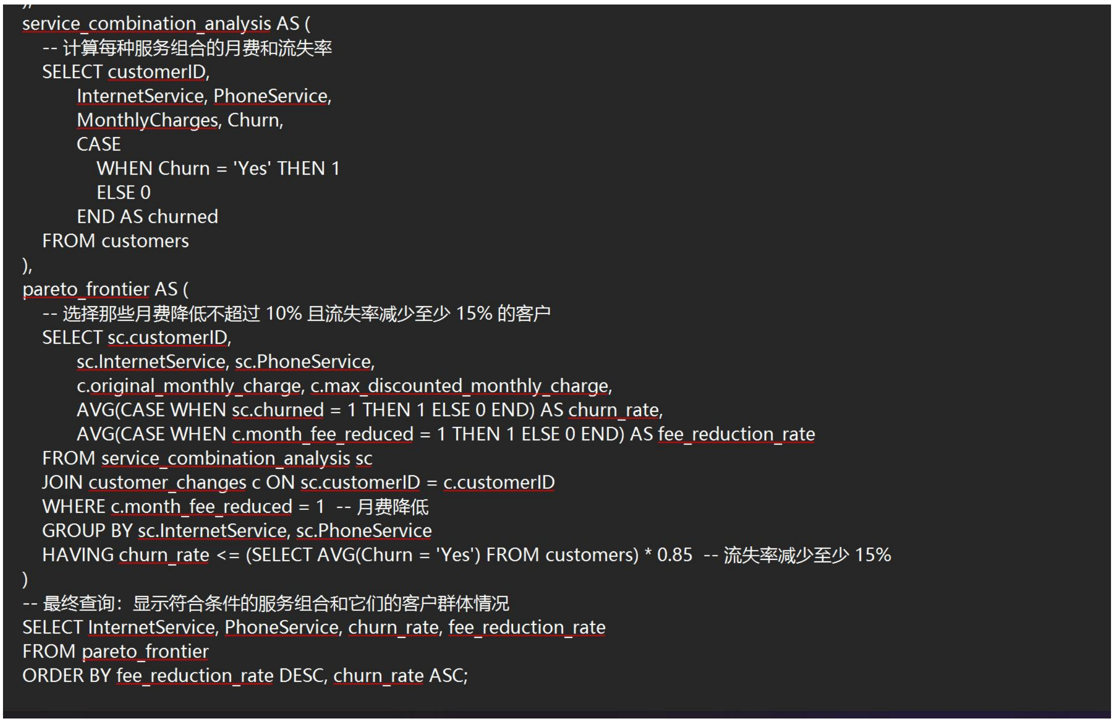


```
fee_trends AS (
    SELECT
        customerID,
        InternetService,
        PhoneService,
        CASE
            WHEN last_month_change > 0 AND two_months_change > 0 THEN 'Consistent Increase'
            WHEN last_month_change < 0 AND two_months_change < 0 THEN 'Consistent Decrease'
            WHEN last_month_change != 0 OR two_months_change != 0 THEN 'Fluctuating'
            ELSE 'Stable'
        END AS trend
    FROM
        fee_history
    WHERE
        last_month_change IS NOT NULL
)
SELECT
    InternetService,
    PhoneService,
    trend,
    COUNT(*) AS count
FROM
    fee_trends
GROUP BY
    InternetService, PhoneService, trend
ORDER BY
    count DESC;
```

第二次找出能使以下两个指标同时优化的客户群体：(1)月费降低不超过 10% (2)流失率减少至少 15%，要求显示各服务组合的帕累托前沿。"



```
WITH customer_changes AS (
  -- 计算每个客户的月费变化和流失情况
  SELECT customerID,
      MonthlyCharges AS original_monthly_charge,
      MonthlyCharges * 0.9 AS max_discounted_monthly_charge,
      Churn,
      CASE
        WHEN Churn = 'Yes' THEN 1
        ELSE 0
      END AS churned,
      CASE
        WHEN MonthlyCharges < (MonthlyCharges * 0.9) THEN 1
        ELSE 0
      END AS month_fee_reduced
  FROM customers
),
service_combination_analysis AS (
  -- 计算每种服务组合的月费和流失率
  SELECT customerID,
      InternetService, PhoneService,
      MonthlyCharges, Churn,
      CASE
        WHEN Churn = 'Yes' THEN 1
        ELSE 0
      END AS churned
  FROM customers
),
```



```
service_combination_analysis AS (
  -- 计算每种服务组合的月费和流失率
  SELECT customerID,
      InternetService, PhoneService,
      MonthlyCharges, Churn,
      CASE
        WHEN Churn = 'Yes' THEN 1
        ELSE 0
      END AS churned
  FROM customers
),
pareto_frontier AS (
  -- 选择那些月费降低不超过 10% 且流失率减少至少 15% 的客户
  SELECT sc.customerID,
      sc.InternetService, sc.PhoneService,
      c.original_monthly_charge, c.max_discounted_monthly_charge,
      AVG(CASE WHEN sc.churned = 1 THEN 1 ELSE 0 END) AS churn_rate,
      AVG(CASE WHEN c.month_fee_reduced = 1 THEN 1 ELSE 0 END) AS fee_reduction_rate
  FROM service_combination_analysis sc
  JOIN customer_changes c ON sc.customerID = c.customerID
  WHERE c.month_fee_reduced = 1  -- 月费降低
  GROUP BY sc.InternetService, sc.PhoneService
  HAVING churn_rate <= (SELECT AVG(Churn = 'Yes') FROM customers) * 0.85  -- 流失率减少至少 15%
)
-- 最终查询：显示符合条件的服务组合和它们的客户群体情况
SELECT InternetService, PhoneService, churn_rate, fee_reduction_rate
FROM pareto_frontier
ORDER BY fee_reduction_rate DESC, churn_rate ASC;
```

问题解析：

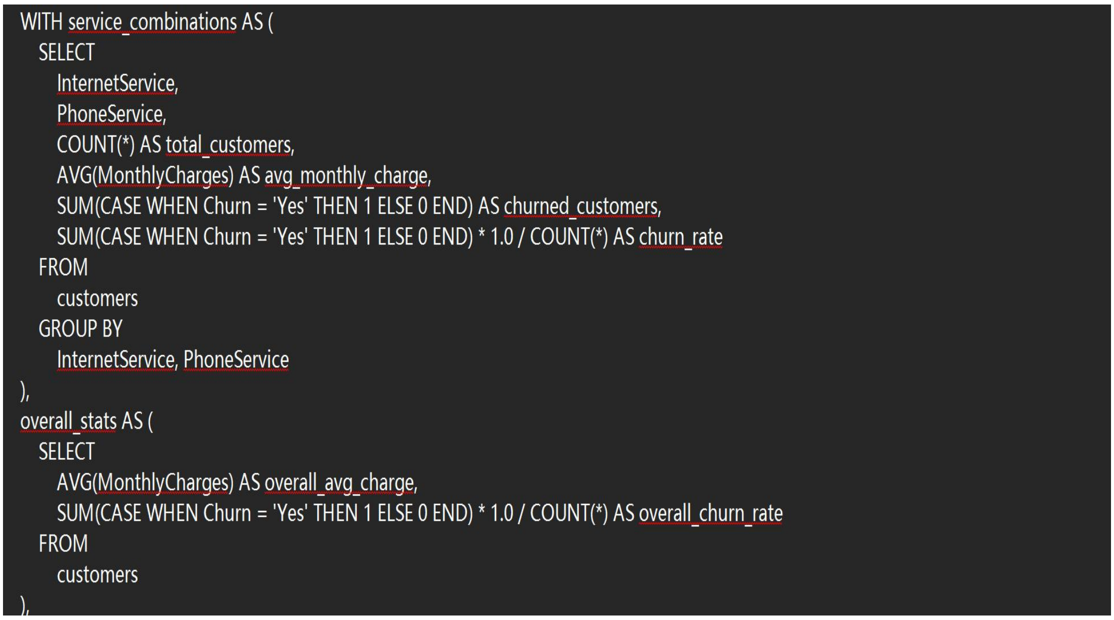
1.条件逻辑错误：CASE WHEN MonthlyCharges < (MonthlyCharges * 0.9) THEN 1 ELSE 0 END

这个条件永远为假（数值不可能小于自身的 90%）

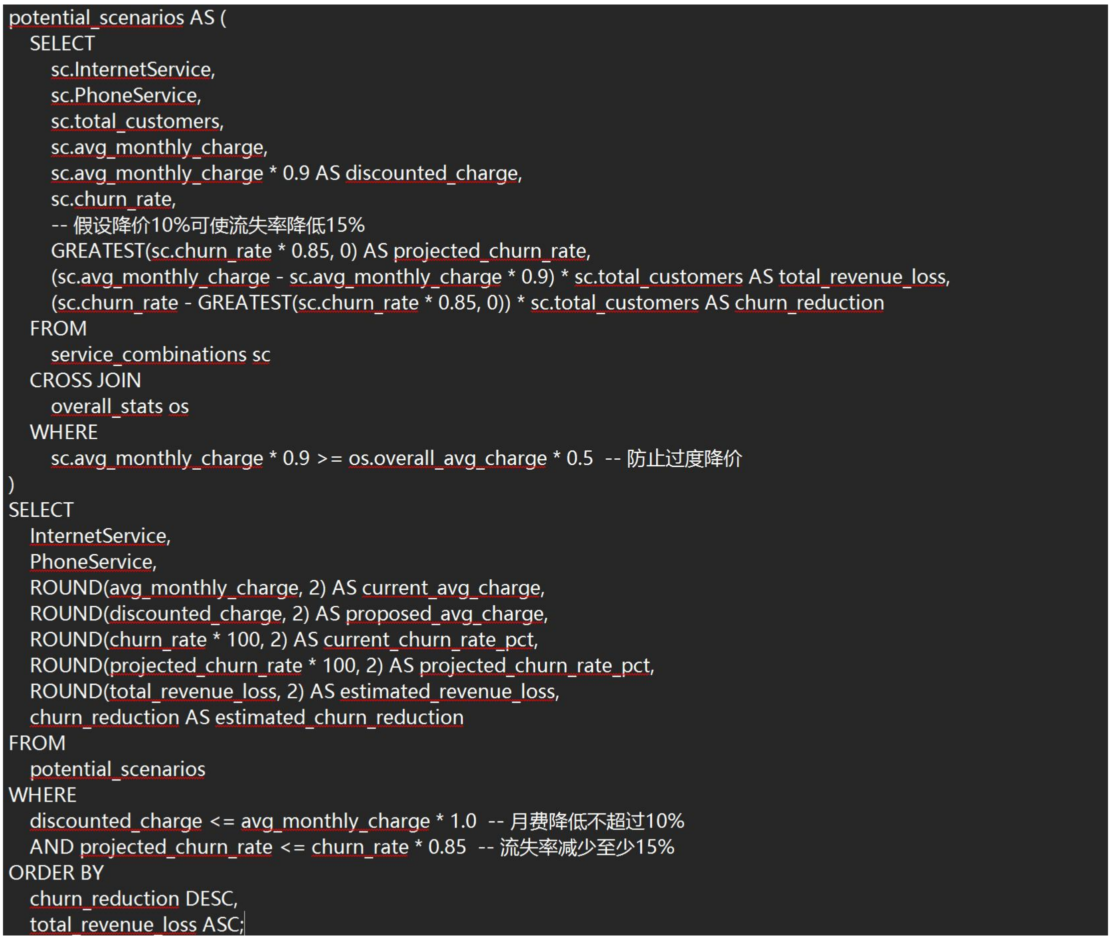
2.帕累托前沿定义问题：当前查询没有真正实现多目标优化，HAVING 条件中的流失率计算方式不正确

3. 数据关联问题； customer_changes 和 service_combination_analysis 包含重复计算，JOIN 条件缺少实际关联逻辑

正确代码：



```
WITH service_combinations AS (
    SELECT
        InternetService,
        PhoneService,
        COUNT(*) AS total_customers,
        AVG(MonthlyCharges) AS avg_monthly_charge,
        SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) AS churned_customers,
        SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 1.0 / COUNT(*) AS churn_rate
    FROM
        customers
    GROUP BY
        InternetService, PhoneService
),
overall_stats AS (
    SELECT
        AVG(MonthlyCharges) AS overall_avg_charge,
        SUM(CASE WHEN Churn = 'Yes' THEN 1 ELSE 0 END) * 1.0 / COUNT(*) AS overall_churn_rate
    FROM
        customers
),
```



```
potential_scenarios AS (
    SELECT
        sc.InternetService,
        sc.PhoneService,
        sc.total_customers,
        sc.avg_monthly_charge,
        sc.avg_monthly_charge * 0.9 AS discounted_charge,
        sc.churn_rate,
        -- 假设降价10%可使流失率降低15%
        GREATEST(sc.churn_rate * 0.85, 0) AS projected_churn_rate,
        (sc.avg_monthly_charge - sc.avg_monthly_charge * 0.9) * sc.total_customers AS total_revenue_loss,
        (sc.churn_rate - GREATEST(sc.churn_rate * 0.85, 0)) * sc.total_customers AS churn_reduction
    FROM
        service_combinations sc
    CROSS JOIN
        overall_stats os
    WHERE
        sc.avg_monthly_charge * 0.9 >= os.overall_avg_charge * 0.5  -- 防止过度降价
)
SELECT
    InternetService,
    PhoneService,
    ROUND(avg_monthly_charge, 2) AS current_avg_charge,
    ROUND(discounted_charge, 2) AS proposed_avg_charge,
    ROUND(churn_rate * 100, 2) AS current_churn_rate_pct,
    ROUND(projected_churn_rate * 100, 2) AS projected_churn_rate_pct,
    ROUND(total_revenue_loss, 2) AS estimated_revenue_loss,
    churn_reduction AS estimated_churn_reduction
FROM
    potential_scenarios
WHERE
    discounted_charge <= avg_monthly_charge * 1.0  -- 月费降低不超过10%
    AND projected_churn_rate <= churn_rate * 0.85  -- 流失率减少至少15%
ORDER BY
    churn_reduction DESC,
    total_revenue_loss ASC;
```

分析：gpt 在处理一些条件很多的问题时候容易错误，有时候是逻辑错误，有时候是条件错误，还出现过凭空捏造数据列的情况，它在接受信息较多的时候容易漏掉或者弄错一些条件。