

# 数据分析岗位技能全攻略

针对新东方太原分校数据分析岗 · 全代码逐句注释版

基于JD拆解 + 实战经验整理 · 磊子 □ · 2026年6月

## □ 目录

---

### □ 目录

#### 1.1 岗位要求拆解

#### 1.2 技能图谱总览

#### 1.3 学习路线建议

### 2.1 SQL 是什么？

#### 2.2 基础查询语法

最基本的结构——SELECT FROM

过滤数据——WHERE

排序——ORDER BY

#### 2.3 聚合函数与分组

常用聚合函数

GROUP BY 分组

#### 2.4 多表连接——JOIN

四种JOIN类型

实际业务场景

#### 2.5 窗口函数（面试必考！）

基本概念

ROW\_NUMBER / RANK / DENSE\_RANK

分组排名——PARTITION BY

累计计算

LAG / LEAD 偏移

#### 2.6 CASE WHEN 条件判断

#### 2.7 SQL 实战面试题

连续登录问题

留存率计算

### 3.1 Excel 在数据分析中的定位

### 3.2 核心快捷键

### 3.3 必会函数

查找引用类

条件统计类

文本处理类

日期类

### 3.4 数据透视表（核心技能）

创建步骤

常用技巧

### 3.5 Power Query（数据清洗利器）

### 4.1 Python 在数据分析中的角色

### 4.2 Pandas 核心操作

安装

数据读取

数据清洗

分组聚合

日期处理

导出

### 4.3 数据可视化（Matplotlib基础）

### 4.4 一个完整分析案例

### 5.1 Tableau 是什么

### 5.2 Tableau 核心概念

拖拽规则：

### 5.3 基础图表制作

柱状图（对比）

折线图（趋势）

饼图（占比）

地图（区域分布）

热力图（交叉分析）

### 5.4 计算字段

基础计算

条件计算

### 5.5 参数控制（关键技能！）

创建参数步骤：

使用参数：

## 5.6 LOD 表达式（面试加分项！）

三种LOD

常见场景

## 5.7 仪表盘制作

布局原则

制作步骤

仪表盘交互功能

## 5.8 数据源管理

数据混合（Data Blending）

数据提取优化

## 6.1 什么是指标体系

## 6.2 OSM 模型（最实用）

OSM实战（教培行业）

## 6.3 指标分级

北极星指标

三级指标体系

## 6.4 指标口径统一

为什么口径重要？

口径文档模板

## 6.5 常用分析框架

漏斗分析

拆解分析

同比环比

## 7.1 监控看板设计

监控看板层级

监控看板必备元素

## 7.2 预警规则设计

常见预警类型

预警阈值设定方法

## 7.3 使用SQL实现监控

## 8.1 为什么要做数据清洗

常见数据问题

## 8.2 SQL 清洗实战

## 8.3 Python 清洗实战

## 9.1 报告类型

## 9.2 报告结构模板

例行报告（周报/月报）模板

## 9.3 报告撰写原则

## 9.4 分析框架示例

问题诊断分析框架

## 10.1 技术面试准备清单

SQL 必会题（面试常考）

Tableau 必会题

数据分析基础

## 10.2 行为面试准备

常见问题

## 10.3 新东方教培行业面试重点

### A.1 SQL 速查表

### A.2 Excel 速查表

### A.3 Python Pandas 速查表

### A.4 Tableau 速查表

# 数据分析岗位技能全攻略·白皮书

针对新东方太原分校数据分析岗位的完整技能体系与实战指南

作者：磊子 ☐ · 2026年6月

## ☐ 目录

第一章：岗位解读与技能地图

第二章：SQL 从入门到实战

第三章：Excel 数据分析实战

第四章：Python 数据分析基础

第五章：Tableau 可视化与仪表盘

第六章：业务指标体系搭建

第七章：数据监控与预警

第八章：数据清洗与处理

第九章：数据分析报告撰写

第十章：面试准备与常见问题

附录：常用命令与函数速查表

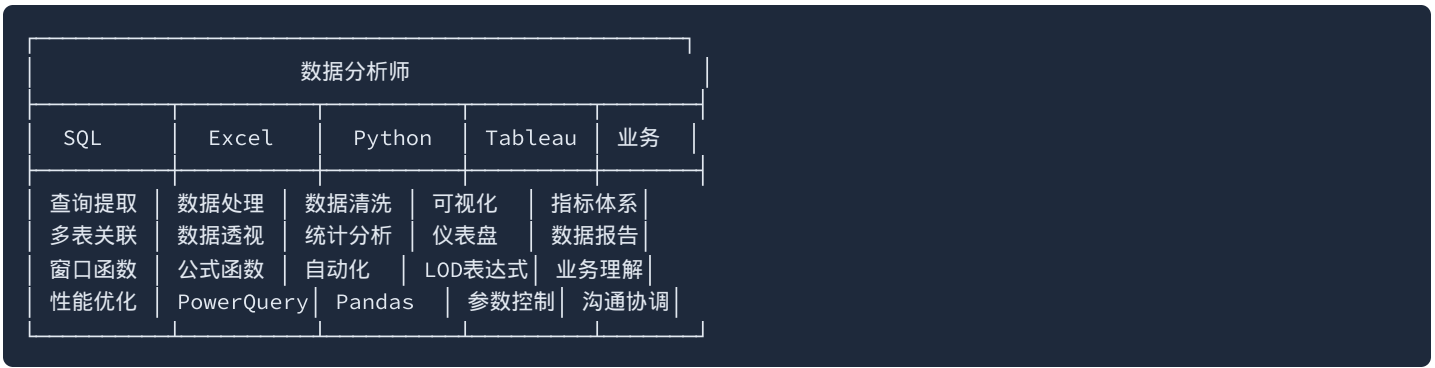
## 第一章：岗位解读与技能地图

### 1.1 岗位要求拆解

根据招聘JD，本岗位的核心职责可以归纳为三个层面：

职责层面	具体内容	对应技能
**数据支持**	搭建可视化看板、完成数据报表	Tableau、SQL、Excel
**数据治理**	指标体系建设、数仓表结构优化	指标体系方法论、数仓知识
**分析报告**	撰写周期性分析报告、推动业务决策	分析框架、业务理解、PPT/报告写作

### 1.2 技能图谱总览



### 1.3 学习路线建议

阶段	学习内容	预计时间
**第一阶段（基础）**	SQL基础增删改查、Excel公式、基础图表	1-2周
**第二阶段（进阶）**	SQL窗口函数、数据透视表、Python Pandas、Tableau基础	2-3周
**第三阶段（高阶）**	指标体系、Tableau LOD、Python数据分析、报告撰写	2-3周
**第四阶段（面试）**	刷SQL题、项目复盘、模拟面试	1周

## 第二章：SQL 从入门到实战

### 2.1 SQL 是什么？

SQL ( Structured Query Language, 结构化查询语言 ) 是用来和数据库"对话"的语言。你可以把它理解为：

"需要你帮我从仓库里找出这些数据，按照这个规则整理好给我。"

数据分析师80%以上的时间都在写SQL，这是最重要的技能。

### 2.2 基础查询语法

#### 最基本的结构——SELECT FROM

```

-- ===== 最基本的SQL查询结构 =====
-- SELECT: 选择你要查看的列(字段)
-- FROM: 指定从哪张表里查
-- *: 通配符, 代表"所有列"
-- 注意: SELECT * 在生产数据库可能影响性能, 大数据量时不推荐

SELECT * FROM orders; -- 查询orders表中所有列的所有数据
-- △ 常见错误: SELECT * 在表字段很多时可能返回超大数据量, 导致查询卡死或网络拥堵
-- △ 建议: 正式SQL中写具体列名, 比如 SELECT id, name FROM orders

-- 查询特定列
SELECT order_id, user_id, amount, create_time
FROM orders;

-- 限制返回行数
SELECT * FROM orders LIMIT 10;

```

### 核心概念解释:

- **SELECT**: 选择你要看的列(字段)
- **FROM**: 从哪个表里找
- **\***: 所有列
- **LIMIT**: 只看前N行(调试时很有用)

### 过滤数据——WHERE

```

-- ===== WHERE: 数据筛选过滤 =====
-- 比较运算符: >(大于) <(小于) >=(大于等于) <=(小于等于)
--              =(等于, 注意是单个等号) !=(不等于)

SELECT * FROM orders
WHERE amount > 100; -- 筛选条件: 只保留金额大于100的行
-- △ 常见错误1: 判断相等用 = 而不是 == (SQL里单个等号就是等于)
-- △ 常见错误2: 字符串和日期要用单引号 '包裹', 数字不用引号
-- △ 常见错误3: NULL不能用 = NULL 判断, 要用 IS NULL 或 IS NOT NULL

-- 多个条件
SELECT * FROM orders
WHERE amount > 100
  AND status = 'completed' -- AND: 同时满足
  AND create_time >= '2026-01-01';

-- 或者条件
SELECT * FROM users
WHERE city = '太原' OR city = '北京'; -- OR: 满足任一

```

### 排序——ORDER BY

```

-- ===== ORDER BY: 排序 =====
-- DESC = descending (降序, 从大到小/从新到旧)
-- ASC = ascending (升序, 从小到大/从旧到新), 不写默认ASC

SELECT order_id, user_id, amount
FROM orders
ORDER BY amount DESC;    -- 按金额从高到低排序, 大单在最前面
-- △ 常见错误1: DESC只作用于它前面那一列, 多列排序每个字段要分别写DESC
--     例: ORDER BY amount DESC, create_time ASC -- 金额降序, 金额相同的按时间升序
-- △ 常见错误2: ORDER BY 中引用的列如果在SELECT中没有, 某些数据库会报错

-- 按日期从旧到新
SELECT * FROM orders
ORDER BY create_time ASC; -- ASC: 升序(小到大), 默认就是ASC

```

## 2.3 聚合函数与分组

### 常用聚合函数

函数	作用	示例
`COUNT()`	计数	`COUNT(DISTINCT user_id)` 去重用户数
`SUM()`	求和	`SUM(amount)` 总金额
`AVG()`	求平均	`AVG(amount)` 平均金额
`MAX()`	最大值	`MAX(amount)` 最大金额
`MIN()`	最小值	`MIN(amount)` 最小金额

### GROUP BY 分组

```

-- ===== GROUP BY: 分组聚合 =====
-- 把数据按指定列分组, 然后对每个组做聚合计算
-- SELECT中出现的非聚合列, 必须出现在GROUP BY中

SELECT city,
       COUNT(*) as user_count -- COUNT(*): 统计每个分组内的行数, 别名user_count
FROM users
GROUP BY city;              -- 按城市分组, 每个城市一行
-- 结果: 太原→120人, 北京→350人, 上海→280人
-- △ 常见错误1: SELECT里写了某列但GROUP BY里没有! 如SELECT city, name FROM users GROUP BY city 会报错
-- △ 常见错误2: COUNT(*)统计所有行, COUNT(列名)统计该列非空行数, 两者结果可能不同
-- △ 常见错误3: GROUP BY放在WHERE之后、ORDER BY之前

-- 每个月的销售额

```

```

SELECT DATE_TRUNC('month', create_time) as month,
       SUM(amount) as total_sales
FROM orders
GROUP BY DATE_TRUNC('month', create_time)
ORDER BY month;

-- HAVING: 对分组结果再次过滤
SELECT city, COUNT(*) as user_count
FROM users
GROUP BY city
HAVING COUNT(*) > 100; -- 只看用户数超过100的城市

```

□ **WHERE vs HAVING 的区别:** **WHERE** 是在分组之前过滤行, **HAVING** 是在分组之后过滤组。比如"只看已完成的订单"用 **WHERE**, "只看订单数超过100的用户"用 **HAVING**。

## 2.4 多表连接——JOIN

实际工作中数据不会都在一个表里, 需要用JOIN把它们连起来。

### 四种JOIN类型

```

-- ===== INNER JOIN: 内连接 (两表交集) =====
-- 表别名: FROM users a 把users表简称为a, 查询时a.user_name
-- INNER JOIN只返回两表匹配上的行, 任一表没有匹配则丢弃

SELECT a.user_name,      -- 从用户表取用户名
       b.order_id,      -- 从订单表取订单ID
       b.amount         -- 从订单表取金额
FROM users a
INNER JOIN orders b ON a.user_id = b.user_id; -- 关联条件: 用户ID相等
-- 结果: 只有下过单的用户才出现, 没下过单的用户不显示
-- △ 常见错误1: 忘记写ON条件, 变成笛卡尔积 (两表所有行互相匹配, 数据量爆炸!)
--   例: users表100行, orders表1000行, 没写ON就变成100×1000=100000行
-- △ 常见错误2: 关联字段类型不一致 (INT vs VARCHAR) 导致匹配不上
-- △ 常见错误3: 关联字段有NULL, NULL和任何值不相等, 匹配不上

-- LEFT JOIN: 左表全部 + 右表匹配的 (没有则为NULL)
SELECT a.user_name, b.order_id, b.amount
FROM users a
LEFT JOIN orders b ON a.user_id = b.user_id;
-- 结果里每个用户都有, 没下过单的用户订单信息为NULL

-- RIGHT JOIN: 右表全部 + 左表匹配的 (很少用)
-- FULL OUTER JOIN: 两表的并集 (很少用)

```

## 实际业务场景

```
-- ===== 业务场景：三表关联统计 =====
-- students(学生) → enrollments(报名记录) → courses(课程)，一对多对多
-- 目标是查出「报了名但没有选任何课」的学生（可能报名了但还没选课）

SELECT
    s.student_id,          -- 学生ID
    s.name,                -- 学生姓名
    s.phone,               -- 联系电话
    COUNT(c.course_id) as course_count, -- 该学生报了多少门课
    SUM(c.amount) as total_paid      -- 该学生已交总金额
FROM students s           -- 学生表（主表，LEFT JOIN保留所有学生）
LEFT JOIN enrollments e ON s.student_id = e.student_id -- 关联报名记录表
LEFT JOIN courses c ON e.course_id = c.course_id      -- 关联课程表
WHERE s.is_active = 1    -- 只看活跃学生（1=活跃）
GROUP BY s.student_id, s.name, s.phone -- 按学生分组
HAVING COUNT(c.course_id) = 0;      -- 筛选出没有选任何课程的学生

-- △ 常见错误1: GROUP BY漏了s.name或s.phone（SELECT中所有非聚合列都必须出现在GROUP BY中）
-- △ 常见错误2: 多表关联时如果有重复字段名，必须用别名前缀区分（如s.student_id和e.student_id）
```

## 2.5 窗口函数（面试必考！）

窗口函数是SQL进阶的核心，也是数据分析面试的高频考点。

### 基本概念

**窗口函数 = 对"每行数据及其周围的数据"进行计算，而不减少行数。**

### ROW\_NUMBER / RANK / DENSE\_RANK

```
-- ===== 窗口函数排名（面试必考！） =====
-- 窗口函数特点：不减少行数，每行保留，同时计算排名
-- OVER()定义"窗口范围"（计算作用域）

SELECT
    student_id,          -- 学生ID
    course_id,           -- 课程ID
    score,               -- 考试分数
    ROW_NUMBER() OVER (ORDER BY score DESC) as row_rank, -- 连续排名：1,2,3,4,5...
    RANK() OVER (ORDER BY score DESC) as rank,          -- 跳跃排名：1,2,2,4,5...
    DENSE_RANK() OVER (ORDER BY score DESC) as dense_rank -- 密集排名：1,2,2,3,4...
FROM exam_scores;
```

```

-- 三个排名的区别（非常重要，面试高频！）：
-- 假设四个学生分数：100, 95, 95, 90
-- ROW_NUMBER(): 1, 2, 3, 4 ← 每人唯一编号，分数相同也分先后（随机）
-- RANK(): 1, 2, 2, 4 ← 分数相同并列，但下一个排名跳号
-- DENSE_RANK(): 1, 2, 2, 3 ← 分数相同并列，下一个排名不跳号

-- △ 常见错误1：窗口函数必须带OVER(), 少了会报错
-- △ 常见错误2：OVER()里忘了ORDER BY, 排名顺序随机
-- △ 常见错误3：ROW_NUMBER()排序字段有重复值时结果不稳定，每次运行排名可能不同

```

### 三个区别（重要！）：

- **ROW\_NUMBER()**：连续不重复（分数相同也给不同编号）
- **RANK()**：跳跃重复（分数相同并列，但下一个跳号）
- **DENSE\_RANK()**：连续重复（分数相同并列，下一个不跳号）

### 分组排名——PARTITION BY

```

-- ===== PARTITION BY: 分组内排名 =====
-- PARTITION BY相当于"在每个组里单独计算", 类似GROUP BY+窗口函数的结合
-- 每个科目内部各有自己的第1名、第2名...

SELECT
    course_id, -- 科目ID (分组维度)
    student_id, -- 学生ID
    score, -- 分数
    ROW_NUMBER() OVER (PARTITION BY course_id ORDER BY score DESC) as course_rank -- 各科内部排名
FROM exam_scores;

-- 结果示例：
-- 数学：小明98→第1, 小红95→第2, 小刚90→第3
-- 英语：小刚96→第1, 小红94→第2, 小明88→第3
-- △ 常见错误1：PARTITION BY和ORDER BY的顺序—先分组再排序
-- △ 常见错误2：忘了PARTITION BY就变成全局排名，不是组内排名了

```

### 累计计算

```

-- ===== 窗口函数：累计求和 =====
-- SUM() OVER(ORDER BY ...): 逐行累加，不是分组求和
-- 第一行=1月销售额，第二行=1+2月，第三行=1+2+3月...

SELECT
    month, -- 月份
    sales_amount, -- 当月销售额
    SUM(sales_amount) OVER (ORDER BY month) as cum_sales -- 从第1个月到当前月的累计
FROM monthly_sales;

-- 结果：
-- 1月 | 100 | 100
-- 2月 | 150 | 250 (100+150)

```

```

-- 3月 | 120 | 370 (100+150+120)
-- △ 常见错误: SUM() OVER()如果忘了ORDER BY, 就变成全局总和, 每行都一样!
-- 错误: SUM(sales_amount) OVER() → 每行都显示全年总和

-- 移动平均(最近3个月)
SELECT
    month,
    sales_amount,
    AVG(sales_amount) OVER (ORDER BY month ROWS BETWEEN 2 PRECEDING AND CURRENT ROW) as moving_avg
FROM monthly_sales;

```

## LAG / LEAD 偏移

```

-- ===== LAG/LEAD: 取前后行数据(环比分析利器) =====
-- LAG(列名, N): 取当前行之前第N行的数据
-- LEAD(列名, N): 取当前行之后第N行的数据

SELECT
    month, -- 月份
    sales_amount, -- 当月销售额
    LAG(sales_amount, 1) OVER (ORDER BY month) as prev_month_sales, -- 上个月的销售额
    -- 环比额 = 当月 - 上月
    sales_amount - LAG(sales_amount, 1) OVER (ORDER BY month) as month_over_month_diff -- 环比增减额
FROM monthly_sales;

-- 结果示例:
-- 2月 | 150 | 100 | 50 (2月比1月多了50)
-- 3月 | 120 | 150 | -30 (3月比2月少了30)
-- △ 常见错误1: 没有ORDER BY时LAG无法确定前一行是什么
-- △ 常见错误2: 第一行没有前一行, LAG返回NULL, 计算结果也是NULL
-- 可以用COALESCE(LAG(...), 0)把NULL转成0

```

## 2.6 CASE WHEN 条件判断

```

-- ===== CASE WHEN: 条件判断(SQL里的if-else) =====
-- 按顺序判断每个WHEN, 遇到第一个满足的就返回对应的THEN
-- 如果都不满足, 返回ELSE的值; 没有ELSE则返回NULL

SELECT
    order_id, -- 订单ID
    amount, -- 金额
    CASE -- 开始条件判断
        WHEN amount < 100 THEN '小额' -- 小于100 → 小额
        WHEN amount >= 100 AND amount < 500 THEN '中额' -- 100-499 → 中额
        WHEN amount >= 500 THEN '大额' -- 500及以上 → 大额
        ELSE '未知' -- 兜底
    END as amount_level -- 结束, 结果列别名amount_level
FROM orders;

-- △ 常见错误1: CASE WHEN的顺序很重要! 如果先写WHEN amount<500再写WHEN amount>100,
-- 那么300会先命中第一个条件(因为300<500)!
-- 正确做法: 把范围小的条件放在前面

```

```

-- △ 常见错误2: 忘记写ELSE, 未覆盖的数据变NULL
-- △ 常见错误3: CASE后直接跟字段名 (CASE amount WHEN 100 THEN...) 和
--     CASE WHEN后写表达式是两种用法, 不要混用

-- 行转列 (经典用法)
SELECT
    month,
    SUM(CASE WHEN city = '太原' THEN amount ELSE 0 END) as taiyuan,
    SUM(CASE WHEN city = '北京' THEN amount ELSE 0 END) as beijing,
    SUM(CASE WHEN city = '上海' THEN amount ELSE 0 END) as shanghai
FROM sales
GROUP BY month;

```

## 2.7 SQL 实战面试题

### 连续登录问题

```

-- ===== 连续登录问题 (面试必考经典题!) =====
-- 核心思路: 用「登录日期减去行号」构造分组键
-- 连续的天数差值相同, 中断了差值就变
-- 例: 1号登录→行号1→差值=0号, 2号登录→行号2→差值=0号 (同组)
--     4号登录→行号3→差值=1号 (差值变了, 说明中断了)

WITH login_dates AS (
    -- CTE 1: 去重, 同一天多次登录只算一次
    SELECT DISTINCT user_id, login_date
    FROM login_log
),
date_ranks AS (
    -- CTE 2: 给每个用户的登录记录编号 (从1开始)
    SELECT
        user_id,
        login_date,
        ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY login_date) as rn
    -- 按用户分组、按日期排序, 生成连续编号
    FROM login_dates
),
diff AS (
    -- CTE 3: 用日期减去编号得到分组键
    SELECT
        user_id,
        login_date,
        DATE_SUB(login_date, INTERVAL rn DAY) as group_key -- 日期减天数=分组标识
    FROM date_ranks
)
-- 最终: 按用户和分组键聚合, 统计连续天数
SELECT user_id,
    COUNT(*) as continuous_days -- 统计这个连续段的天数
FROM diff
GROUP BY user_id, group_key -- 按用户+连续段分组
HAVING COUNT(*) >= 7; -- 只保留连续7天及以上的
-- △ 常见错误1: DATE_SUB是MySQL函数。SQL Server用DATEADD(day,-rn,date), PostgreSQL用date - rn

```

```
-- △ 常见错误2: 忘记去重 (第一步DISTINCT), 同一天多次登录会导致编号错位
-- △ 常见错误3: PARTITION BY user_id漏了, 变成全局编号, 所有用户混在一起算
```

## 留存率计算

```
-- ===== 留存率计算 (面试必考!) =====
-- 留存率 = 第N天还登录的用户数 / 新增用户数
-- 次日留存: 第一天来了, 第二天又来了的比例
-- 7日留存: 第一天来了, 第7天又来了的比例

WITH first_login AS (
  -- CTE 1: 找出每个用户的首次登录日期 (新增日)
  SELECT
    user_id,
    MIN(login_date) as first_date  -- 最早登录日期就是新增日
  FROM login_log
  GROUP BY user_id
),
retention AS (
  -- CTE 2: 按新增日统计留存情况
  SELECT
    f.first_date,  -- 新增日期
    COUNT(DISTINCT l.user_id) as new_users,  -- 当天新增用户数
    COUNT(DISTINCT CASE WHEN DATEDIFF(l.login_date, f.first_date) = 1
      THEN l.user_id END) as day1_retained,  -- 次日留存的用户数
    COUNT(DISTINCT CASE WHEN DATEDIFF(l.login_date, f.first_date) = 6
      THEN l.user_id END) as day7_retained  -- 7日留存的用户数
  FROM first_login f
  LEFT JOIN login_log l ON f.user_id = l.user_id  -- LEFT JOIN保留所有新增用户 (包括流失的)
  GROUP BY f.first_date
)
SELECT
  first_date,  -- 新增日期
  new_users,  -- 新增用户数
  ROUND(day1_retained / new_users * 100, 2) as day1_retention_rate,  -- 次日留存率 (百分比, 保留2位小数)
  ROUND(day7_retained / new_users * 100, 2) as day7_retention_rate  -- 7日留存率
FROM retention
ORDER BY first_date;

-- △ 常见错误1: DATEDIFF (大日期, 小日期)的顺序—要确保后续登录日减首次登录日是正数
-- △ 常见错误2: 没有用LEFT JOIN, 只计算了有留存用户的留存率 (漏掉了流失用户, 新用户数偏小)
-- △ 常见错误3: COUNT(DISTINCT)比COUNT(*)慢, 但这里必须用DISTINCT (同一天多次登录)
```

## 第三章: Excel 数据分析实战

### 3.1 Excel 在数据分析中的定位

虽然SQL和Python能处理更大的数据量，但Excel仍然是数据分析师最常用的工具之一，特别是：

- **快速查看数据**：打开CSV/Excel就能看
- **临时分析**：不需要写代码，拖拽就能出结果
- **汇报呈现**：很多业务方和管理层习惯看Excel表

## 3.2 核心快捷键

快捷键	作用	说明
`Ctrl + Shift + ↓`	选中到数据末尾	快速选择整列数据
`Ctrl + T`	创建超级表	数据会自动扩展，公式自动填充
`Alt + N + V`	创建数据透视表	最快的方法
`Ctrl + Shift + L`	打开/关闭筛选	筛选器开关
`F4`	切换引用方式	绝对引用/混合引用切换
`Ctrl + `	显示所有公式	调试时非常有用

## 3.3 必会函数

### 查找引用类

```
VLOOKUP(lookup_value, table_array, col_index_num, [range_lookup])  
|-- 找什么 --|-- 在哪里找 --|-- 返回第几列 --|-- 精确/模糊 --
```

```
' ===== VLOOKUP 查找引用（最常用函数之一） =====  
' VLOOKUP(找什么, 在哪找, 返回第几列, 是否精确匹配)  
' A2 - 你要查的值（比如学生姓名）  
' $B$2:$D$100 - 查找范围, $是绝对引用（拖拽公式时不会变）  
' 3 - 返回范围中第3列的值  
' FALSE - 精确匹配（必须写FALSE!）  
' △ 常见错误1: 第四个参数忘了写FALSE, 默认为TRUE（模糊匹配）, 经常返回错误值  
' △ 常见错误2: 查找值必须在范围的第一列—VLOOKUP只能从左往右查！  
' △ 常见错误3: $B$2:$D$100 没加$变成相对引用, 下拉公式时范围会跑偏  
=VLOOKUP(A2, $B$2:$D$100, 3, FALSE)  
  
' ===== XLOOKUP (VLOOKUP的升级版, Excel 2021+/Office 365) =====  
' 不需要查找列在第1列, 可以返回任意列  
' 查找列: B:B, 返回列: C:C
```

```
' △ XLOOKUP比VLOOKUP更好用，但老版Excel不支持
=XLOOKUP(A2, B:B, C:C)
```

### ### 条件统计类

```
' ===== 条件统计函数 =====
```

```
' SUMIF：单条件求和——对满足条件的单元格求和
```

```
' 参数：条件区域, 条件, 求和区域
```

```
=SUMIF(A:A, "太原", B:B) ' 统计A列中为"太原"的对应的B列总和
```

```
' SUMIFS：多条件求和——满足多个条件的单元格求和
```

```
' 注意！SUMIFS的参数顺序是(求和区域, 条件区域1, 条件1, ...)
```

```
' △ SUMIF和SUMIFS的参数顺序不同！SUMIF是(条件区域, 条件, 求和区域)
```

```
=SUMIFS(求和区域, 条件区域1, 条件1, 条件区域2, 条件2)
```

```
' COUNTIF：单条件计数
```

```
=COUNTIF(A:A, "太原") ' A列中"太原"出现几次
```

```
' COUNTIFS：多条件计数
```

```
=COUNTIFS(A:A, "太原", B:B, "已报名") ' A列=太原且B列=已报名的个数
```

```
' AVERAGEIF：条件求平均
```

```
' △ 注意：条件中的文本要用双引号括起来，数字不需要
```

```
=AVERAGEIF(条件区域, 条件, 求平均区域)
```

```
' 实战示例：统计太原校区已报名人数
```

```
=COUNTIFS(A:A, "太原", B:B, "已报名")
```

```
' △ 常见错误：条件写错了引号——数字不用引号(">100"正确)，文本必须引号("太原"正确)
```

## 文本处理类

```
' ===== 文本处理函数 =====
```

```
=LEFT(A1, 3) ' 取A1左边3个字符 - "1381234"→"138"
```

```
=RIGHT(A1, 4) ' 取A1右边4个字符 - "1381234"→"1234"
```

```
=MID(A1, 2, 3) ' 从第2位开始取3个字符 - "ABCDE"→"BCD"
```

```
=LEN(A1) ' 返回文本长度 - "Hello"→5
```

```
' △ 注意：LEN(中文)在Excel中是一个字符算1，不是字节数
```

```
=FIND("太原", A1) ' 返回"太原"在A1中出现的位置，找不到返回#VALUE！
```

```
=CONCATENATE(A1, B1) ' 合并A1和B1的内容
```

```
' △ 更简单的合并方式是用 & 符号: =A1 & B1
=TEXT(A1, "yyyy-mm-dd")      ' 把数字/日期按指定格式显示
=IFERROR(A1/B1, "错误")      ' 如果公式出错 (如除零), 返回"错误"而不是#DIV/0!
' △ IFERROR建议每次做除法都包一层, 防止碰到空单元格导致的除零错误

### 日期类
```

' ===== 日期函数 =====

=TODAY() ' 返回今天的日期 (自动更新)

' △ 想要固定日期用快捷键 Ctrl+; (加分号)

=YEAR(A1) ' 从日期中提取年份 — 2026-06-14 → 2026

=MONTH(A1) ' 提取月份 — 2026-06-14 → 6

=DATEDIF(A1, B1, "d") ' 计算A1到B1相差的天数

' DATEDIF的第三个参数: "d"=天数, "m"=整月数, "y"=整年数

' △ DATEDIF是Excel的隐藏函数, 输入时没有自动提示, 但可以直接用

=WEEKDAY(A1, 2) ' 返回周几, 参数2表示周一=1, 周二=2...周日=7

' △ WEEKDAY的第二参数很重要: 1 (默认) =周日=1, 2=周一=1, 3=周一=0

## 3.4 数据透视表 (核心技能)

### 创建步骤

1. 选中数据区域 → 插入 → 数据透视表
2. 把字段拖到四个区域:

区域	作用	示例
**行标签**	纵向展示的维度	校区、月份
**列标签**	横向展示的维度	课程类型
**值**	要统计的指标	报名人数、金额
**筛选器**	对整个透视表筛选	年份

### 常用技巧

- **值字段设置**: 右键值 → 值字段设置 → 可以改计算方式 (求和/计数/平均值) 和显示方式 (占比/差异)
- **分组**: 右键日期 → 组合 → 按月/季度/年
- **切片器**: 选中透视表 → 插入切片器 → 做成交互式筛选按钮
- **计算字段**: 选中透视表 → 分析 → 字段、项目和集 → 计算字段

## 3.5 Power Query (数据清洗利器)

---

Power Query 是 Excel 内置的数据清洗工具，适合做重复性的数据整理工作。

### 常用操作：

- 删除空行/空列
  - 拆分列 (按分隔符)
  - 合并查询 (类似SQL的JOIN)
  - 分组依据 (类似SQL的GROUP BY)
  - 添加自定义列 (类似Excel公式)
- 

## 第四章：Python 数据分析基础

### 4.1 Python 在数据分析中的角色

---

Python不是必需的，但会了会是巨大的加分项。主要用在：

- **自动化处理**：每天/每周自动跑报表
- **大数据量**：Excel打不开的几百万行数据
- **统计分析**：假设检验、回归分析
- **机器学习**：预测模型 (加分项，非必需)

### 4.2 Pandas 核心操作

---

#### 安装

```
# 安装Python数据分析常用库
# pandas: 数据处理 (核心库, 必装)
# openpyxl: 读写Excel文件
# matplotlib: 数据可视化 (画图)
pip install pandas openpyxl matplotlib
# △ 如果提示pip命令找不到, 试试 pip3 install ...
# △ 如果安装慢, 加国内镜像: pip install -i https://pypi.tuna.tsinghua.edu.cn/simple ...
```

## 数据读取

```
import pandas as pd # 导入pandas库,起别名pd(约定俗成)

# 读取CSV文件
# read_csv 会自动识别表头,第一行变成列名
df = pd.read_csv('data.csv')
# △ 如果CSV文件编码是GBK(中文Windows常见),要加 encoding='gbk'

# 读取Excel文件(可以指定sheet页)
df = pd.read_excel('data.xlsx', sheet_name='Sheet1')

# 从数据库读取(需要先安装sqlalchemy和pymysql)
from sqlalchemy import create_engine
engine = create_engine('mysql+pymysql://用户名:密码@主机地址/数据库名')
df = pd.read_sql('SELECT * FROM orders', engine)

# 快速查看数据(这4行是必杀技,拿到数据先跑一遍)
df.head() # 显示前5行,快速了解数据结构
df.info() # 显示每列的数据类型、非空数量、内存占用
df.describe() # 数值列的统计摘要(均值、标准差、四分位数等)
df.shape # 返回(行数,列数),比如(10000,15)
# △ 注意:df.head()默认显示5行,想看更多行用 df.head(10)
```

## 数据清洗

```
# 查看缺失值
df.isnull().sum()

# 删除缺失值
df = df.dropna(subset=['重要列']) # 删掉重要列为空的行
df = df.dropna(thresh=len(df.columns)-3) # 删掉缺失超过3列的行

# 填充缺失值
df['列名'] = df['列名'].fillna(0) # 填0
df['列名'] = df['列名'].fillna(df['列名'].mean()) # 填平均值
df['列名'] = df['列名'].fillna(method='ffill') # 填上一个值

# 删除重复行
df = df.drop_duplicates()
df = df.drop_duplicates(subset=['user_id']) # 按某列去重

# 修改数据类型
df['日期'] = pd.to_datetime(df['日期'])
df['金额'] = df['金额'].astype(float)

# 筛选
high_value = df[df['金额'] > 1000] # 类似SQL的WHERE
beijing = df[df['城市'] == '北京']
condition = df[(df['金额'] > 500) & (df['城市'] == '太原')]
```

```
# 排序
df_sorted = df.sort_values('金额', ascending=False) # 降序
```

## 分组聚合

```
# 按城市分组统计 (类似SQL: GROUP BY)
result = df.groupby('城市')['金额'].agg(['sum', 'mean', 'count', 'max'])
result.columns = ['总金额', '平均金额', '订单数', '最大金额']

# 多维度分组
result = df.groupby(['城市', '课程类型'])['金额'].sum()

# 透视表 (Pivot Table)
pivot = pd.pivot_table(
    df,
    values='金额',
    index='城市',
    columns='课程类型',
    aggfunc='sum',
    fill_value=0
)
```

## 日期处理

```
# 提取日期成分
df['年份'] = df['日期'].dt.year
df['月份'] = df['日期'].dt.month
df['周几'] = df['日期'].dt.dayofweek # 0=周一
df['周'] = df['日期'].dt.isocalendar().week

# 按月份统计
monthly = df.set_index('日期').resample('M')['金额'].sum()
```

## 导出

```
df.to_csv('result.csv', index=False, encoding='utf-8-sig')
df.to_excel('result.xlsx', index=False, sheet_name='报表')
```

## 4.3 数据可视化 (Matplotlib基础)

---

```

import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.sans-serif'] = ['SimHei'] # 支持中文
matplotlib.rcParams['axes.unicode_minus'] = False

# 折线图 (趋势)
plt.figure(figsize=(10, 5))
plt.plot(df['月份'], df['销售额'], marker='o')
plt.title('月度销售额趋势')
plt.xlabel('月份')
plt.ylabel('销售额 (万元)')
plt.grid(True)
plt.savefig('trend.png', dpi=150, bbox_inches='tight')
plt.show()

# 柱状图 (对比)
plt.figure(figsize=(10, 5))
plt.bar(df['城市'], df['销售额'], color='steelblue')
plt.title('各城市销售额对比')
plt.xticks(rotation=45)
plt.savefig('bar.png', dpi=150, bbox_inches='tight')

# 饼图 (占比)
plt.figure(figsize=(8, 8))
sizes = df['占比']
labels = df['类别']
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title('各类别占比')
plt.savefig('pie.png', dpi=150, bbox_inches='tight')

```

## 4.4 一个完整分析案例

```

import pandas as pd
import matplotlib.pyplot as plt

# 1. 读取数据
df = pd.read_excel('教学数据.xlsx')

# 2. 数据清洗
df = df.drop_duplicates()
df = df.dropna(subset=['学生ID'])
df['报名日期'] = pd.to_datetime(df['报名日期'])

# 3. 分析: 每月报名人数趋势
monthly = df.set_index('报名日期').resample('M').size()

# 4. 分析: 各课程报名人数
course_stats = df.groupby('课程名称').agg({
    '学生ID': 'count',
    '缴费金额': 'sum'
}).rename(columns={'学生ID': '报名人数', '缴费金额': '总收入'})

# 5. 分析: 按校区统计
campus_stats = df.pivot_table(

```

```
values='缴费金额',
index='校区',
columns='课程名称',
aggfunc='sum',
fill_value=0
)

# 6. 导出结果
with pd.ExcelWriter('分析结果.xlsx') as writer:
    monthly.to_excel(writer, sheet_name='月度趋势')
    course_stats.to_excel(writer, sheet_name='课程统计')
    campus_stats.to_excel(writer, sheet_name='校区课程收入')

print("分析完成！结果已保存到 分析结果.xlsx")
```

## 第五章：Tableau 可视化与仪表盘

### 5.1 Tableau 是什么

Tableau 是目前最主流的数据可视化工具之一，也是招聘JD明确要求的技能。它的核心优势是：

- **拖拽式操作**：不需要写代码
- **交互式仪表盘**：点击某个区域，其他图表自动联动
- **美观的图表**：秒杀Excel自带的图表
- **企业级看板**：可以发布到服务器，多人共用

### 5.2 Tableau 核心概念

概念	英文	解释
<b>**维度**</b>	Dimension	分类数据（蓝色），如城市、课程、月份
<b>**度量**</b>	Measure	数值数据（绿色），如销售额、人数
<b>**工作表**</b>	Worksheet	单个图表
<b>**仪表盘**</b>	Dashboard	多个工作表的组合
<b>**故事**</b>	Story	多个仪表盘按顺序讲述一个故事
<b>**工作簿**</b>	Workbook	包含所有工作表、仪表盘的文件（.twb或.twbx）
<b>**数据提取**</b>	Extract (.hyper)	将数据拉到Tableau本地，比实时连接快

**实时连接**	Live Connection	直接连接数据库实时查询
----------	-----------------	-------------

## 拖拽规则：

维度（蓝色）放到行/列 → 度量（绿色）放到行/列 → 就出图了

- 蓝色字段拖到行/列：生成表头/标签
- 绿色字段拖到行/列：生成坐标轴/数值
- 度量拖到颜色/大小/标签：给图表上色、缩放、加标签

## 5.3 基础图表制作

---

### 柱状图（对比）

3. 把"城市"（维度）拖到**列**
4. 把"销售额"（度量）拖到**行**
5. 点击工具栏"智能推荐"，选柱状图

### 折线图（趋势）

6. 把"订单日期"（维度）拖到**列**，右键选"连续的月份"
7. 把"销售额"（度量）拖到**行**
8. 自动生成折线图

### 饼图（占比）

9. 把"课程分类"拖到**列**
0. 把"报名人数"拖到**行**
1. 标记类型改为"饼图"
2. 把"课程分类"再拖到"颜色"
3. 把"报名人数"拖到"标签"

### 地图（区域分布）

4. 双击"省份"（地理角色已分配）
5. 自动生成地图
6. 把"销售额"拖到颜色

## 热力图（交叉分析）

7. 把"城市"拖到行
8. 把"课程类型"拖到列
9. 把"销售额"拖到颜色
0. 颜色编码自动显示高低

## 5.4 计算字段

---

计算字段是Tableau的核心进阶技能，相当于在Tableau里写公式。

### 基础计算

```
-- 计算字段名：利润率  
SUM([利润]) / SUM([销售额])  
  
-- 计算字段名：折扣金额  
[原价] * [折扣率]  
  
-- 计算字段名：同比  
(SUM([2026年销售额]) - SUM([2025年销售额])) / SUM([2025年销售额])
```

### 条件计算

```
-- 高价值客户标记  
IF SUM([消费金额]) > 10000 THEN "VIP"  
ELSEIF SUM([消费金额]) > 5000 THEN "重要"  
ELSE "普通"  
END
```

## 5.5 参数控制（关键技能！）

---

参数让用户通过下拉菜单或滑块动态切换指标。这是面试高频考点。

### 创建参数步骤：

1. 在数据窗格右键 → 创建 → 参数
2. 设置参数名 (如"选择指标")
3. 数据类型: 字符串
4. 允许的值: 列表 → 输入"销售额, 利润, 订单数"

### 使用参数:

5. 创建计算字段:

```
CASE [选择指标] -- 参数名
  WHEN "销售额" THEN SUM([销售额])
  WHEN "利润" THEN SUM([利润])
  WHEN "订单数" THEN COUNT([订单ID])
END
```

6. 把这个计算字段拖到视图中
7. 右键参数 → 显示参数控件 → 出现下拉菜单

## 5.6 LOD 表达式 (面试加分项!)

LOD (Level of Detail) 是Tableau最强大的功能之一, 允许你控制计算的"粒度".

### 三种LOD

类型	语法	作用
**FIXED**	{FIXED [维度]: 聚合表达式}	忽略视图中的所有维度
**INCLUDE**	{INCLUDE [维度]: 聚合表达式}	在视图维度的基础上额外增加维度
**EXCLUDE**	{EXCLUDE [维度]: 聚合表达式}	排除视图中的某个维度

### 常见场景

```
-- 每个客户的总消费 (忽略视图筛选)
{FIXED [客户ID] : SUM([消费金额])}

-- 对比每个城市的销售额占总销售额的百分比
SUM([销售额]) / {FIXED : SUM([销售额])}

-- 每个订单金额占客户总消费的比例
SUM([订单金额]) / {FIXED [客户ID] : SUM([订单金额])}
```

## 5.7 仪表盘制作

---

### 布局原则

8. **上重点下辅助**：最重要的KPI放在左上角
9. **左到右、上到下**：遵循用户阅读习惯
0. **一致性**：颜色、字体、样式统一
1. **10秒原则**：用户10秒内应该看懂仪表盘的核心信息

### 制作步骤

2. 创建2-5个工作表（每个一张图表）
3. 点击底部"新建仪表盘"
4. 从左侧拖入工作表到仪表盘区域
5. 添加筛选器、参数控件
6. 设置图表联动：仪表盘 → 操作 → 添加操作 → "筛选器"
7. 调整布局：使用水平/垂直容器实现自适应
8. 添加标题、说明文字

### 仪表盘交互功能

功能	操作	效果
**图表筛选器**	右键图表 → 用作筛选器	点击一个图表，其他图表联动
**参数控件**	右键参数 → 显示参数控件	下拉菜单/滑块切换指标
**URL动作**	仪表盘 → 操作 → 转到URL	点击跳转到详情页面
**工具提示**	工具提示卡里拖入更多字段	鼠标悬停显示详细信息
**高亮**	仪表盘 → 添加 → 突出显示动作	同颜色数据一起高亮

## 5.8 数据源管理

---

### 数据混合 (Data Blending)

当主数据源和次要数据源无法直接连接时使用。

- 9. 数据 → 新数据源 → 添加第二个数据源
- 0. 分析 → 创建混合关系 → 选择一个公共字段连接
- 1. 次要数据源的字段使用时，会自动按公共字段匹配

## 数据提取优化

- 大数据集 (>100万行) 使用数据提取 (.hyper)
- 提取时做聚合过滤：仅提取需要的行和列
- 定期刷新：通过 Tableau Server 或命令行刷新

# 第六章：业务指标体系搭建

## 6.1 什么是指标体系

指标体系就是一套有层次、有关联的指标集合，用来全面衡量业务健康度。

"没有指标，业务好不好就是凭感觉；有指标体系，一切用数据说话。"

## 6.2 OSM 模型（最实用）

数分面试中OSM模型是必考题，也是实际工作中最实用的方法论。

字母	英文	中文	例子（教育培训行业）
**O**	Objective	业务目标	提升课程报名转化率
**S**	Strategy	业务策略	优化课程体验课、推出限时优惠
**M**	Measurement	衡量指标	体验课预约率、试听转化率、优惠券核销率

## OSM实战（教培行业）

Objective（战略目标）：扩大太原市场占有率

- Strategy 1: 提升品牌知名度
  - Measurement: 品牌搜索量、公众号关注数
- Strategy 2: 提升招生效率
  - Measurement: 获客成本、线索转化率、到店率
- Strategy 3: 提升续费率
  - Measurement: 课程完成率、家长满意度、续费意愿

## 6.3 指标分级

### 北极星指标

公司最重要的那个指标，所有工作都围绕它展开。

- 教培行业北极星指标示例：**有效学员数** 或 **续报率**
- 电商行业北极星指标：**GMV**（成交总额）
- 互联网产品北极星指标：**DAU**（日活跃用户）

### 三级指标体系

北极星指标：有效学员数

↓

一级指标（全局层面）：

- 新招学员数
- 学员流失数
- 学员续费率

↓

二级指标（业务层面）：

- 市场：曝光量、线索量、获客成本
- 销售：接通率、邀约率、转化率
- 教学：出勤率、作业提交率、满意度
- 服务：投诉率、续费率、转介绍率

↓

三级指标（执行层面）：

- 每日电话拨打量、接通量
- 体验课预约数、到店人数
- 各科产出出勤明细
- 每节课的评分明细

## 6.4 指标口径统一

## 为什么口径重要？

同一个指标，不同部门理解不同，会导致数据打架。

### 例子——“转化率”可能有多种定义：

- 销售部：签单人数 / 试听到店人数
- 市场部：试听预约数 / 广告曝光量
- 管理层：付费学员 / 所有线索

## 口径文档模板

【指标名】课程报名转化率  
【业务定义】在体验课结束后的7天内购买正式课程的学生比例  
【计算公式】 $\text{购买正式课程人数} / \text{参加体验课人数} \times 100\%$   
【统计周期】按周/月度统计  
【数据来源】报名系统 体验课记录表 + 缴费记录表  
【更新频率】每日更新  
【负责部门】运营部  
【备注】排除内部员工子女、免费名额

## 6.5 常用分析框架

### 漏斗分析

曝光线索 → 到店试听 → 购买课程 → 完课续费



- **发现瓶颈：**哪一步转化率最低？
- **优化方向：**针对瓶颈环节做改进

### 拆解分析

总营收 = 学员数 × 客单价 × 购课频次

年度购课次数

哪个因子下降最多，就是问题所在。

## 同比环比

- **同比**：跟去年同期比（消除季节性因素）
- **环比**：跟上个月/上周比（看近期趋势）

```
-- ===== 环比计算（月环比） =====
-- 环比 = 跟上个月比，公式：(本月-上月)/上月×100%

SELECT
    month,                                -- 月份
    sales,                                  -- 本月销售额
    LAG(sales, 1) OVER (ORDER BY month) as prev_month, -- 上个月的数据
    -- 环比增长率（保留1位小数）
    ROUND((sales - LAG(sales, 1) OVER (ORDER BY month)) /
          LAG(sales, 1) OVER (ORDER BY month) * 100, 1) as mom_growth
FROM monthly_sales;

-- △ 常见错误1：分母为0会报错！可用NULLIF(分母, 0)预防
--   安全写法：sales / NULLIF(LAG(sales,1) OVER(), 0)
-- △ 常见错误2：第1行没有前一行数据，计算结果为NULL，这是正常的

-- 同比
SELECT
    month,
    sales,
    LAG(sales, 12) OVER (ORDER BY month) as same_month_last_year,
    ROUND((sales - LAG(sales, 12) OVER (ORDER BY month)) /
          LAG(sales, 12) OVER (ORDER BY month) * 100, 1) as yoy_growth
FROM monthly_sales;
```

## 第七章：数据监控与预警

### 7.1 监控看板设计

数据监控是数据分析师的重要日常职责。

## 监控看板层级

层级	受众	刷新频率	重点内容
**战略层**	总经理、VP	日报/周报	北极星指标、营收、学员数
**管理层**	部门负责人	日报	部门KPI完成进度
**执行层**	一线员工	实时/小时	个人指标、待跟进事项

## 监控看板必备元素

5. **KPI卡片**：关键数字突出展示，带同比/环比变化
6. **趋势图**：最近N天/周的变化趋势
7. **预警标识**：异常值标红/闪烁
8. **下钻功能**：从总体下钻到明细

## 7.2 预警规则设计

---

### 常见预警类型

类型	判定规则	示例
**阈值预警**	指标超过/低于设定阈值	转化率低于10%触发预警
**波动预警**	环比/同比波动超过N%	日营收环比下降超20%
**趋势预警**	连续N天下降/上升	连续3天新增线索下降
**数据质量预警**	缺失率异常、数据延迟	昨日数据凌晨3点还未到齐

### 预警阈值设定方法

9. **历史数据法**：计算过去30天的均值 $\pm$ 2倍标准差
0. **业务经验法**：由业务方根据经验设定
1. **对比法**：与行业平均水平对比

## 7.3 使用SQL实现监控

---

```

-- ===== 每日监控预警SQL (实战常用!) =====
-- 带预警标记的监控查询, 自动标出异常

SELECT
    report_date, -- 统计日期
    new_students, -- 当天新增学生数
    LAG(new_students, 1) OVER (ORDER BY report_date) as prev_day, -- 前一天的新增数
    LAG(new_students, 7) OVER (ORDER BY report_date) as prev_week_same_day, -- 上周同一天
    -- 环比变动百分比 (用NULLIF防除0)
    ROUND((new_students - LAG(new_students, 1) OVER (ORDER BY report_date)) /
        NULLIF(LAG(new_students, 1) OVER (ORDER BY report_date), 0) * 100, 1) as mom_change_pct,
    -- 预警逻辑: 自动标出异常
    CASE
        WHEN new_students < LAG(new_students, 1) OVER (ORDER BY report_date) * 0.7 THEN '△ 大幅下降'
        -- 比前一天下降超过30%
        WHEN new_students < LAG(new_students, 7) OVER (ORDER BY report_date) * 0.8 THEN '△ 低于上周同期'
        -- 比上周同一天下降超过20%
        WHEN new_students IS NULL THEN '⊠ 数据缺失'
        -- 数据还没到
        ELSE '⊠ 正常'
        -- 正常范围
    END as alert_flag
FROM daily_student_stats
WHERE report_date >= CURRENT_DATE - INTERVAL 30 DAY -- 最近30天
ORDER BY report_date DESC; -- 最新日期排前面

-- △ 常见错误1: NULLIF(前一日数据, 0)如果前一日为0就转NULL, 避免除零报错
-- △ 常见错误2: CASE WHEN按顺序判断, 第一个满足就退出, 条件顺序重要
-- △ 常见错误3: LAG(..., 7)需要连续7天数据, 中间缺日期可能不准

```

## 第八章：数据清洗与处理

### 8.1 为什么要做数据清洗

真实的数据永远是不完美的。数据清洗是数据分析的第一步，也往往是最耗时的步骤。

#### 常见数据问题

问题	示例	影响
缺失值	电话号码为空	无法统计有效线索
重复值	同一学员重复录入	统计数据虚高
异常值	年龄200岁、金额为负数	影响平均值计算

格式不一致	手机号有些带-有些不带	无法合并匹配
逻辑错误	报名日期晚于毕业日期	数据不可信

## 8.2 SQL 清洗实战

```

-- ===== 数据清洗：清理手机号 =====
-- 把手机号中的-、空格等字符全部去掉，只保留数字

UPDATE students
SET phone = REGEXP_REPLACE(phone, '[^0-9]', '') -- 将非数字字符替换为空
WHERE phone IS NOT NULL; -- 只处理非空的手机号
-- 结果: "138-1234-5678" -> "13812345678"
-- △ 安全提醒: UPDATE是危险操作! 强烈建议先改成SELECT确认效果!
-- 安全步骤:
-- SELECT phone, REGEXP_REPLACE(phone, '[^0-9]', '')
-- FROM students WHERE phone IS NOT NULL;
-- △ 常见错误: 忘记写WHERE, 会把所有phone更新, 包括空值变空字符串
-- △ REGEXP_REPLACE函数名在不同数据库不同: MySQL用REGEXP_REPLACE, PostgreSQL用REGEXP_REPLACE, SQL Server用dbo.RegexReplace

-- 标记异常数据
SELECT *,
CASE
WHEN age < 0 OR age > 100 THEN '异常'
WHEN age BETWEEN 0 AND 3 THEN '需确认'
ELSE '正常'
END as data_quality
FROM students;

-- 处理重复数据 (保留最早的记录)
WITH numbered AS (
SELECT *,
ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY create_time ASC) as rn
FROM students
)
DELETE FROM students WHERE (user_id, create_time) IN (
SELECT user_id, create_time FROM numbered WHERE rn > 1
);

```

## 8.3 Python 清洗实战

```

import pandas as pd

df = pd.read_csv('raw_data.csv')

# 1. 去重
df = df.drop_duplicates(subset=['手机号'])

```

```
# 2. 处理缺失值
df['年龄'] = df['年龄'].fillna(df['年龄'].median()) # 年龄填中位数
df['邮箱'] = df['邮箱'].fillna('未知') # 文本填默认值
df = df.dropna(subset=['姓名', '手机号']) # 关键信息缺失则删除

# 3. 处理异常值
df = df[df['年龄'].between(0, 100)]
df = df[df['金额'] > 0]

# 4. 格式统一
df['手机号'] = df['手机号'].astype(str).str.replace(r'\D', '', regex=True)
df['日期'] = pd.to_datetime(df['日期'], errors='coerce')

# 5. 检查清洗效果
print(f"清洗前: {len(df)}行")
print(f"清洗后: {len(df)}行")
print(f"删除: {5000 - len(df)}行异常数据")
print(f"缺失值统计: {df.isnull().sum()}")
```

## 第九章：数据分析报告撰写

### 9.1 报告类型

类型	频率	受众	内容重点
**日报**	每日	运营/管理层	昨日核心指标、异常预警
**周报**	每周	管理层	本周趋势、重点项目进展
**月报**	每月	部门/公司领导	月度回顾、KPI达成、问题分析
**专题报告**	按需	相关方	深度分析一个具体问题

### 9.2 报告结构模板

#### 例行报告（周报/月报）模板

【标题】XX校区20XX年X月运营分析月报

##### 一、核心数据概览

- 本月核心KPI（附同比/环比变化）
- KPI达成率（目标 vs 实际）

└ 一句话总结 ("本月整体表现平稳,续费率有所提升")

## 二、详细分析

- └ 招生分析: 各渠道表现、获客成本
- └ 教学分析: 出勤率、完课率、满意度
- └ 财务分析: 营收、利润、人效

## 三、异常与预警

- └ 数据异常点 (标注原因)
- └ 风险提示 (需重点关注的事项)

## 四、改进建议

- └ 数据层面的发现
- └ 可执行的行动建议

## 五、附录

- └ 数据口径说明、原始数据表

## 9.3 报告撰写原则

原则	说明	反面例子	正面例子
**结论先行**	先说结果再说过程	"我们做了A/B/C...最后发现X提升"	"X指标提升20%, 主要因为..."
**数据说话**	每句话都有数据支撑	"最近招生不错"	"本月招生120人, 环比增长15%"
**可视化**	能用图就不用表, 能用表就不用文字	一大段文字描述趋势	折线图+一句话总结
**可执行**	报告不能只分析, 要有行动建议	"转化率下降了"	"建议优化体验课流程, 预计可提升5%转化率"

## 9.4 分析框架示例

### 问题诊断分析框架

问题: 本月续费率从60%下降到45%

Step 1: 数据验证

- 确认数据无误, 排除口径/统计问题

Step 2: 维度拆解

- 按校区、年级、课程、教师拆解
- 发现: 初三毕业班续费率下降最明显 (从55%→20%)

#### Step 3: 原因定位

- 中考政策变化, 毕业班学生不再需要课外辅导
- 属于行业周期性现象, 非教学质量问题

#### Step 4: 策略建议

- 1. 毕业班推出"暑假预科班", 衔接高中课程
- 2. 针对非毕业班加强家长沟通, 稳住基本盘

#### Step 5: 效果追踪

- 下个月继续监控续费率变化

---

## 第十章：面试准备与常见问题

### 10.1 技术面试准备清单

---

#### SQL 必会题（面试常考）

2. 窗口函数：ROW\_NUMBER、RANK、DENSE\_RANK的区别
3. 连续N天登录的用户
4. 留存率计算（次日/7日/30日）
5. 行转列/列转行
6. TopN问题（每个分类取前3名）
7. 中位数计算
8. 同比环比计算
9. 累计求和/移动平均

#### Tableau 必会题

0. 参数的作用和使用方法
1. LOD表达式：FIXED/INCLUDE/EXCLUDE的区别
2. 数据混合和数据连接的区别
3. 如何实现仪表盘交互联动
4. 大数据量时的性能优化

#### 数据分析基础

5. 什么是指标体系？怎么搭建？
6. 留存率怎么算？

- 7. A/B测试的基本原理
- 8. 如何判断两个数据是否有显著差异

## 10.2 行为面试准备

---

### 常见问题

#### Q: "数据结论和业务直觉冲突时怎么办?"

回答思路：先确认数据口径和计算过程是否正确 → 如果数据正确，不要怀疑数据 → 和业务方深入沟通，理解"直觉"背后的经验 → 找出差异原因，可能是数据口径不同或业务方经验在新的环境下失效 → 综合双方意见，达成共识

#### Q: "描述一个你推动业务决策的数据分析项目"

回答思路：使用STAR原则

- Situation：业务背景，遇到了什么问题

- Task：你的分析目标是什么

- Action：你用到了什么数据、什么工具、什么分析方法

- Result：你的分析结论是什么，业务方采纳了什么建议，带来了什么效果

## 10.3 新东方教培行业面试重点

---

针对新东方太原分校的特点，面试可能会重点考察：

- 9. **教培行业核心指标**：续报率、满班率、退费率、转介绍率
- 0. **招生漏斗分析**：曝光 → 咨询 → 试听 → 报名 → 续费
- 1. **季节性分析**：寒暑假高峰期、开学后的策略
- 2. **校区对比**：不同校区的运营效率对比
- 3. **教师效能分析**：排课率、班均人数、好评率

---

## 附录：常用命令与函数速查表

## A.1 SQL 速查表

```
-- =====  
-- SQL常用命令速查表（快速查阅）  
-- =====  
  
-- ===== 查询基础 =====  
SELECT col1, col2 FROM table WHERE condition;    -- 选取特定列，条件过滤  
SELECT * FROM table LIMIT 10;                   -- 取所有列的前10行（调试用）  
SELECT DISTINCT col FROM table;                 -- 去重，看某列有哪些不同值  
  
-- ===== 聚合函数 =====  
COUNT(*),      -- 计数（包括NULL行）  
SUM(col),       -- 求和  
AVG(col),       -- 求平均  
MAX(col),       -- 最大值  
MIN(col)        -- 最小值  
-- 聚合函数忽略NULL值（COUNT(*)除外）  
SELECT col, COUNT(*) FROM table GROUP BY col HAVING COUNT(*) > N; -- 分组后过滤  
  
-- ===== 多表连接 =====  
INNER JOIN      -- 交集，两表都有  
LEFT JOIN       -- 左表全保留  
RIGHT JOIN      -- 右表全保留（通常用LEFT JOIN互换）  
FULL OUTER JOIN -- 全集（较少用）  
  
-- ===== 窗口函数 =====  
ROW_NUMBER() OVER (PARTITION BY dim ORDER BY measure DESC) as rn -- 组内排名  
RANK() / DENSE_RANK() -- RANK跳号，DENSE_RANK不跳号  
SUM() OVER (ORDER BY date ROWS BETWEEN N PRECEDING AND CURRENT ROW) -- 移动累计  
LAG(col, N) OVER (ORDER BY date) -- 取前N行  
LEAD(col, N) OVER (ORDER BY date) -- 取后N行  
  
-- ===== 条件判断 =====  
CASE WHEN 条件1 THEN 值1 WHEN 条件2 THEN 值2 ELSE 默认值 END -- 顺序判断  
  
-- ===== 日期函数 =====  
DATE_TRUNC('month', date) -- 截断到月初（PostgreSQL）  
DATEDIFF(end, start) -- 日期差（MySQL）  
DATE_ADD(date, INTERVAL N DAY) -- 加N天（MySQL）  
EXTRACT(YEAR FROM date) -- 提取年（PostgreSQL）  
-- 不同数据库日期函数差异很大，注意区分！  
  
-- ===== 字符串函数 =====  
CONCAT(a, b) -- 拼接。△ 有NULL则整个结果为NULL！  
SUBSTRING(str, start, len) -- 截取子串  
REPLACE(str, old, new) -- 替换  
UPPER(str), LOWER(str) -- 大小写转换  
LENGTH(str) -- 长度。△ MySQL返回字节数，CHAR_LENGTH返回字符数！  
TRIM(str) -- 去首尾空格
```

## A.2 Excel 速查表

---

```
' 查找引用
=VLOOKUP(找什么, 在哪找, 返回第几列, FALSE)
=XLOOKUP(找什么, 在哪找, 返回什么)
=INDEX(区域, 行号, 列号)
=MATCH(找什么, 在哪找, 0)

' 条件统计
=SUMIF(条件区域, 条件, 求和区域)
=SUMIFS(求和区域, 条件区域1, 条件1, ...)
=COUNTIF(条件区域, 条件)
=COUNTIFS(条件区域1, 条件1, ...)

' 文本
=LEFT(text, n), RIGHT(text, n), MID(text, start, n)
=LEN(text)
=FIND(find_text, within_text)
=TEXT(value, "yyyy-mm-dd")

' 日期
=TODAY(), =NOW()
=YEAR(date), =MONTH(date), =DAY(date)
=DATEDIF(start, end, "d") ' d/m/y差异

' 逻辑
=IF(条件, 真值, 假值)
=IFERROR(公式, 出错返回)
=AND(条件1, 条件2), =OR(条件1, 条件2)

' 快捷键
Ctrl+T 创建超级表
Alt+N+V 创建数据透视表
Ctrl+Shift+L 筛选开关
F4 切换绝对引用
```

## A.3 Python Pandas 速查表

---

```
import pandas as pd

# 读写
df = pd.read_csv('file.csv')
df = pd.read_excel('file.xlsx', sheet_name='Sheet1')
df.to_csv('out.csv', index=False)
df.to_excel('out.xlsx', index=False)

# 查看
df.head(), df.info(), df.describe(), df.shape, df.columns

# 筛选
df[df['col'] > 100]
```

```

df[(df['col1'] > 100) & (df['col2'] == 'A')]

# 分组
df.groupby('col')['value'].agg(['sum', 'mean', 'count'])
df.pivot_table(values='value', index='row', columns='col', aggfunc='sum')

# 清洗
df.dropna(), df.fillna(0), df.drop_duplicates()
df['col'] = df['col'].astype(int)
df['date'] = pd.to_datetime(df['date'])

# 日期
df['date'].dt.year, .dt.month, .dt.day
df.set_index('date').resample('M')['value'].sum()

# 合并
pd.merge(df1, df2, on='key', how='left')
pd.concat([df1, df2])

```

## A.4 Tableau 速查表

功能	操作/语法
创建表计算	右键度量 → 快速表计算
参数	右键数据窗格 → 创建 → 参数
计算字段	右键数据窗格 → 创建 → 计算字段
LOD FIXED	{FIXED [维度]: SUM([度量])}
LOD INCLUDE	{INCLUDE [维度]: SUM([度量])}
LOD EXCLUDE	{EXCLUDE [维度]: SUM([度量])}
仪表盘联动	仪表盘 → 操作 → 添加 → 筛选器
发布到Server	服务器 → 发布工作簿

### 最后的话

>

数据分析师的核心不是工具或代码，而是用数据回答业务问题的能力。

>

工具（SQL/Excel/Python/Tableau）可以学，但业务敏感度和逻辑思维能力需要在实战中慢慢积累。

>

祝你面试顺利！□

— 全文完 —

祝你面试顺利 □