**分析方法：**

**分析指标（度量）**

分析指标

数据库表中的数值字段可以聚合成原子指标

原子指标也叫基础指标表达业务实体原子量化属性，不可再分的概念集合

派生指标：基础指标加修饰词

衍生指标也叫复合指标

数理衍生： 比率，最大 最小 平均 累计

时间衍生：同比 环比 累计

进度衍生： 完成计划比

 ---------1.原子指标---------

SELECT SUM(A) FROM Z GROUP BY C;

SELECT SUM(B) FROM Z GROUP BY C;

---------2.派生指标---------

SELECT SUM(A) AS SUM\_A FROM Z WHERE D = 'X' GROUP BY C;

SELECT SUM(B) AS SUM\_B FROM Z WHERE D = 'X' AND E = 'Y' GROUP BY C;

---------3.衍生指标---------

SELECT SUM\_A/SUM\_B FROM TEMP;

**分析方法**

单度量（指标）

和维度无关：指标分布，标准值（警戒值）。

单维度：纵向（时间）折线， 横向（维度）条柱形，比例分析饼图，累计分析-二八法则帕累托图（单位个数与份额做累计图），

多维度：多色多符号折线，多族柱形图，雷达图。

多度量（指标）：

两度量相关散点图，波士顿矩阵，双轴折线图。多度量用相关矩阵，矩阵散点图，

杜邦指标分解（正影响度指标）。

多维度（包含索引维度）：多色多符号相关散点图。 多行多列组合折线柱形图

ggplot作图原理：维度和度量都算一列，x轴，y轴各一个列，以后一个特征一个维度。

**MODELER:**

读取excel中日期格式要现在excel中设置号日期，否则在modeler中很难再认定日期。Modeler中的数据类型是自动判断的。

**POWERBI：**

**日期表：**

1.制作日期表，将需要的月度季度年度维度建立，日期表日期设置为日期格式

2.设置日期表

3.将数据表日期与日期表关联

设置为日期表后所有系统不再为任意表的日期列生成层级关系。自己再日期表中建立日期层级关系。涉及日期字段筛选切片均采用日期表日期字段

日期轴受筛选期控制 通过日期智能函数突破日期筛选限制

**度量值的移动：**

1.在菜单栏中选择主表移动

2.在数据模型视图中同时选择多个度量值然后输入显示文件夹

3.在数据模型视图中直接拖动度量值到某个文件夹

**python脚本获取数据到pbi中处理：**

主页选择“获取数据/更多…”，随后依次选择“其它/Python脚本”，点击确定按钮，在其他编辑器中测试好以后粘贴到pbi的编辑窗口，运行后通过导航选择df，或者是运行python.excute()查询后选择表右键添加到新查询。

Python生成多个数据框，如果导航中选择多个数据表，生成多个新查询，都不会让各表同源，多次执行python脚本，效率低。如果想多个表同源。

可以用：源=初始查询名称。或者在初始查询上点右键-选择“引用”。选择某个步骤右键选择“提取之前的步骤”，会生成一个独立查询，现有查询会用提取出来的独立查询作为唯一的源头。即源=提取出来的步骤查询名。多个数据表用一个python源，python源必须要选择导航，可以先导航到表，再删除python源中的导航步骤。

对于有小时分秒粒度的日期时间格式，无法系统设置声明日期表。建立时期时间表，与其他表的日期时间建立关系即可。

**对于查询中间数据集运用python处理。**

选中查询后选择运行python脚本，对dataset使用python处理即可。python有多少个df，处理结果就会呈现多少个表，通过右键来选择df。继续pbi的后续处理。

**SUM,SUMX函数区别**

SUM语法：= SUM（<列名>）

示例：总销售额= SUM（销售表[销售额]）

SUMX语法：= SUMX（<Table>，<expression>）

示例：总销售额SUMX = SUMX（销售表，销售表[数量] \*销售表[单位价格]）

先逐行计算出结果再相加，适用于平均单价，平均压力这种加权平均数。

每次访问平均花费= DIVIDE（[总花费SUMX]， [总访问量]）

**可视化：**

**ANACONDA 与R 结合：**

1. 正常安装R和Rstudio，安装anaconda不要选择注册path变量。
2. 给R注册环境变量path
3. 在电脑里自己下载的r（rstudio）中的Console中输入：

install.packages('IRkernel')

4. 随后在电脑里找到这个包的地址，我的是 "F:/R-3.6.2/library/IRkernel"

然后打开anaconda prompt，输入：

jupyter kernelspec install F:/R-3.6.2/library/IRkernel #安装R内核

5. 接着还是在anaconda prompt里输入：R，回车，接着输入：

IRkernel::installspec(user = FALSE) #注册R内核

6. pip install rpy2 不要用conda，会安装一个新内核，带来麻烦。

7.%load\_ext rpy2.ipython #jupyter调用R命令

**8.spyder和jupyter中用R的编码问题。**

前两者的编码大多用utf-8，但是R一般会跟中文操作系统用GBK。

在R代码开始中加入Sys.setlocale('LC\_ALL', locale = "English\_United States.1252")将中文编码换成utf-8。以防R发出的中文编码提示，在python界面里无法解码显示。？是否一定？

R在读取sqlserver数据库时，中文采用gbk编码，如果想在spyder，jupyter打印或者传出数据时，会被错误解码。因此要转码。

fenqushuiliang$fName <- iconv(as.vector(fenqushuiliang$fName), 'GB2312', 'UTF-8')

注意Encoding(someX)<-'UTF-8'，只是告诉R用什么编码来解码这个变量，并非改变编码。

iconv是R的转码命令。最多只能转向量。不能转dataframe

%R 当前行切换R语言模式

%%R 当前cell切换R语言模式

-o data\_out # jupyter的R输出数据

-i data #向R输入数据

**9.Spyder中的操作：**

import rpy2.robjects as robjects

robjects.r(r\_script)

Spyder的R转出dataframe用array当中介。

matrix = robjects.r['fenqushuiliang']#将dataframe变量通过运行变量名调出来

a = np.array(matrix)

**MATPLOTLIB:**

matplotlib.pyplot.plot()参数详解 plt.plot([x], y, [fmt]= '[color][marker][line]', [x2], y2, [fmt2], ..., \*\*kwargs)

实例：plt.plot(x,y2,color='green', marker='o', linestyle='dashed', linewidth=1, markersize=6)

各个绘图类型参数不同函数名称不同，比如plt.scatter()增加点大小的s参数

**绘图原理**

1.函数命令式作图：

通过plt.plot或者plt.subplot(2，2，1)，定义ax布局和激活当前ax。当参数小于10可以不用逗号；

plt.plot()自动生成fig和ax，多次运行也只作用在同一个绘图区。plt.close（）会在一次命令执行中删除画布，让下一个画图命令新开画板。

作图命令直接作用在当前激活ax绘图区上。命令多次运行在同一个ax绘图区画多个图。命令执行完figure自动回收，无法再调用，命令一次运行更新一次内存。

同时显示多个画布，一定要用plt.figrue()创建新画布，并将新建画布作为当前画布。plt.plot或plt.subplot(221)运行在当前激活的画布的ax上。

多块画布中一块画布的重复使用通过plt.figure（1）给数字编号进行重复调用。如果新建画布没有指定编号，则按照建立顺序作为数字编号来复用。多个axes通过plt.csa(ax1)来切换复用，首先还是要定义ax对象。

命令式样调整

plt. xlim() 限制数轴范围。.axis([xmin，xmax，ymin，ymax]）同时指定xylim

Plt.xticks(ticks=刻度数值列表，labels=刻度名列表，fontsize=字体大小,,rotation=0) ，另有title() xlabel()

plt.legend()#除'best'，另外loc属性有：'upper right', 'upper left', 'lower left', 'lower right', 'right', 'center left', 'center right', 'lower center', 'upper center', 'center'

2.面向对象式做图：

手动创建fig和axes对象，图必须用ax的方法绘制.创建的图形可以重复调用.

fig和axes可以用fig, axes = plt.subplots（figsize=（，），nrows=2, ncols=2，sharey=True，facecolor=“y”，#sharex和sharey：表示坐标轴的属性是否相同，可选的参数：True，False，row，col，默认值均为False，表示画布中的四个ax是相互独立的；

gridspec\_kw=dict(width\_ratios=[4, 3])）

#gridspec\_kw:字典类型，可选参数。把字典的关键字传递给GridSpec构造函数创建子图放在网格里(grid)。

单子图不用加nrows，ncols参数，一次生成画布尺寸和子图布局,选择某一个子图绘图则通过axes[1,1]来选择，单列只需一个位置参数，单子图不需要加位置参数，也可以用ax0,ax1=axes.flatten()。也可以一个一个ax创建，fig=plt.figure(figsize=(a, b), dpi=dpi),figsize 设置图形的大小，a 为图形的宽， b 为图形的高，单位为英寸,dpi 为设置图形每英寸的点数

生成图画对象后用ax1=fig.add\_subplot(211)来创建布局并指定一个一个ax。

fig.suptitle（“我是画布的标题”，fontsize=20）

fig.text（0.45，0.9，“画布副标题”）

fig.subplots\_adjust(left=0.2,bottom=0.1,right=0.8,top=0.8,hspace=0.5)。以fig左下角为原点，建立直角坐标系，0.2表示整个fig的20%。hspace子图的高度间距，wspace子图的宽度间隙，数字大小是子图的比例。

添加嵌套图fig.add\_axes([left以fig的左边缘开始的fig比例,bottom以fig下缘开始的fig的比例,width子图的宽,height子图的高]，facecolor=“y”)。此方法的函数方法plt.axes（[]）

画图ax.plot()等等

ax.set\_xlim（） ax.set\_xticks() ax.set\_xticklabels(),ax.set\_title(),ax.set\_xlabel(),ax.legend(loc='best')

fig.legend(scatterpoints=1,labels = Label\_Com, labelspacing=0.4,columnspacing=0.4,markerscale=2,bbox\_to\_anchor=(0.9, 0),ncol=12,prop=font1,handletextpad=0.1)

bbox\_to\_anchor=(0.9, 0) 通过调节此参数放置图例，(0.9,0) 分别是横向相对位置和纵向

相对位置

scatterpoints=1：设置图例中对应图像只出现一次

labels= Label\_Com ：设置图例中的名称

labelspacing=0.4 调整图例中标签的距离

columnspacing=0.4 调整图例中不同列之间的间距

handletextpad=0.1 用此参数调节图例和标签之间的距离

图例的字体格式在prop中进行设置，赋值font1可以是一个字典，包含各个属性及其对应值，属性包括family（字体）、size（字体大小）等常用属性，更详细的解释可参考matplotlib手册中关于legend prop的解释。一种比较简单的设置为：

font1 = {'family' : 'Times New Roman',

'weight' : 'normal',

'size' : 23,

}

#一次定义

ax.set(xlim=(0, 24), ylabel="",

xlabel="Automobile collisions per billion miles")

设置数轴百分数

from matplotlib import ticker

ax.yaxis.set\_major\_formatter(xmax=100#扩大缩小倍数, decimals=None, symbol='%', is\_latex=False)

ax.spines['right'].set\_visible(False) 不要右边的边框

ax.axes.get\_yaxis().set\_visible(False) 不要Y轴

ax1.axis[:].set\_visible(False) 不要所有的轴

plt.show（）只显示当前内存中的所有图像，不能重复调用。

函数式与面向对象画图也可以混合用，函数式画图默认作用在当前激活的ax上.

函数式显性转对象用ax=plt. gca()获取当前ax对象，fig=gcf()获取当前画布对象

plt. plot(xyz三维图)scatte() bar()contourf()热图，每个图画类型的参数不尽相同需要单独查询。

pie（z输入的数组，labels每一块的类别名，explode每一块的间距，autopct数据输出。格式如“%1.1f”则输出不含百分比的数值。格式如“%1.1f%%”则连接百分号）

Matplotlib可以无缝的处理LaTex字体，在图中加入数学公式

# 添加数学公式和坐标轴标签

plt.text(0.5 \* (a+b), 1, r"$\int\_a^b f(x)\mathrm{d}x$",

horizontalalignment='center', fontsize=20)

# 前两个参数是放置文本的坐标

plt.figtext(0.9, 0.075,'$x$')

plt.figtext(0.075,0.9,'$f(x)$' )

行内公式用 $...$ 也有用\(...\),让公式处于文字行内。

行间公式用\[...\]也有用$$...$$,让公式处于单独一行。

**#自定义颜色**

col = np.where(x<1,'k',np.where(y<5,'b','r'))

plt.scatter(x, y, c=col, s=5, linewidth=0)

**#十字箭头坐标**

1.创建画布并引入axisartist工具。

import mpl\_toolkits.axisartist as axisartist

#创建画布

fig = plt.figure(figsize=(8, 8))

#使用axisartist.Subplot方法创建一个绘图区对象ax

ax = axisartist.Subplot(fig, 111)

#将绘图区对象添加到画布中

fig.add\_axes(ax)

2.绘制带箭头的x-y坐标轴

#通过set\_visible方法设置绘图区所有坐标轴隐藏

ax.axis[:].set\_visible(False)

#ax.new\_floating\_axis代表添加新的坐标轴

ax.axis["x"] = ax.new\_floating\_axis(0,0)

#给x坐标轴加上箭头

ax.axis["x"].set\_axisline\_style("->", size = 1.0)

#添加y坐标轴，且加上箭头

ax.axis["y"] = ax.new\_floating\_axis(1,0)

ax.axis["y"].set\_axisline\_style("-|>", size = 1.0)

#设置x、y轴上刻度显示方向

ax.axis["x"].set\_axis\_direction("top")

ax.axis["y"].set\_axis\_direction("right")

我们先把原始的如上图的所有坐标轴隐藏，即长方形的四个边。

然后用ax.new\_floating\_axis在绘图区添加坐标轴x、y，这里的ax.new\_floating\_axis(0,0)，第一个0代表平行直线，第二个0代表该直线经过0点。同样，ax.axis["y"] = ax.new\_floating\_axis(1,0)，则代表竖直曲线且经过0点。

再次，x.axis["x"].set\_axisline\_style("->", size = 1.0)表示给x轴加上箭头，"->"表示是空箭头，size = 1.0表示箭头大小。ax.axis["y"].set\_axisline\_style("-|>", size = 1.0)中"-|>"则是实心箭头。

最后，设置x、y轴上刻度显示方向，对于x轴是刻度标签在上面还是下面，y轴则是刻度标签在左边还是右边。

3.在带箭头的x-y坐标轴背景下，绘制函数图像

#生成x步长为0.1的列表数据

x = np.arange(-15,15,0.1)

#生成sigmiod形式的y数据

y=1/(1+np.exp(-x))

#设置x、y坐标轴的范围

plt.xlim(-12,12)

plt.ylim(-1, 1)

#绘制图形

plt.plot(x,y, c='b')

**Pyecharts：**

formatter=JsCode("""function(params) {return Math.abs(params.data).toFixed(2);} """ )

#params是根据js下echarts中series中的字典式的变量名来确定的。data表示一个数据而非数组。如果有嵌套则继续选择属性。比如.data.value。data通用value

#params.dataIndex 表示数据位置序号。条件判断params.value> 0 ? ('+' + params.value) : ('-' + params.value); formatter={value}也可以

#对于坐标轴的axislabel\_opts设置 ：params 直接传刻度值

**SEABORN :**

#sns专用分面函数作图

g = sns.FacetGrid(zone3,col="分公司",hue="月份", height=10)

g.map(sns.kdeplot, "累计产销差率")

#类别画图函数可以直接分面。细项绘图函数不行。

g=sns.displot(data=zone3 , x="累计产销差率", hue="月份",col="分公司" ,kind="kde",col\_wrap=2,height=5,aspect=1.5)

g.set\_axis\_labels('累计产销差率','二级分区数量密度')

g.set(ylabel='二级分区集中度')

#细项绘图函数没有尺寸设置？细项绘图函数分面用matplot

f, axes = plt.subplots(2,2, figsize=(12, 10),dpi=100)

ax1,ax2,ax3,ax4=axes.flatten()

#必须用细项函数才可以写到axes里面,一个图不加ax索引，单行列图只加一个索引。

sns.histplot(data=zone3.query("分公司 == '中区'") , x="累计产销差率", bins=10,

hue="月份",multiple='dodge',shrink=.8,kde=True,ax=ax1)

ax1.set\_xticks([xticks])#刻度位置

ax1.set\_xticklabels([xlabel])#刻度文字位置

ax1.set\_xlabel('')#坐标轴名称

ax1.set\_ylabel('二级分区数量')

ax1.set\_title('中区')

#绘制直方图bins可以按照分界点。shrink可以控制控柱子之间的间隙。Kde可以加分布曲线。

g=sns.histplot(data=zone3 , x="累计产销差率", bins=[-20,-10,0,10,20,30,40,50],hue="月份",multiple='dodge',shrink=.8,kde=True)

sns.despine(left=True) #移除边框

**VIZ社交网络图：**