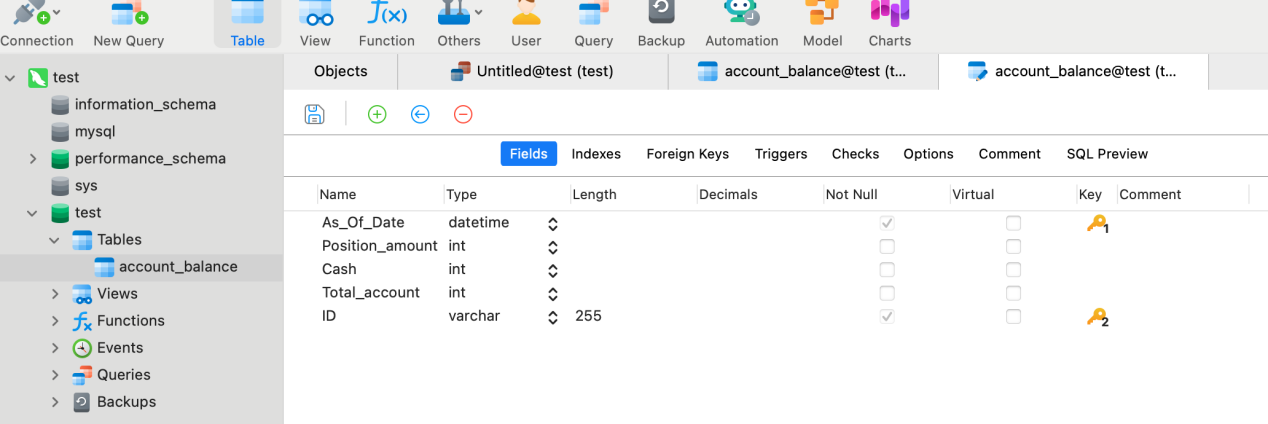
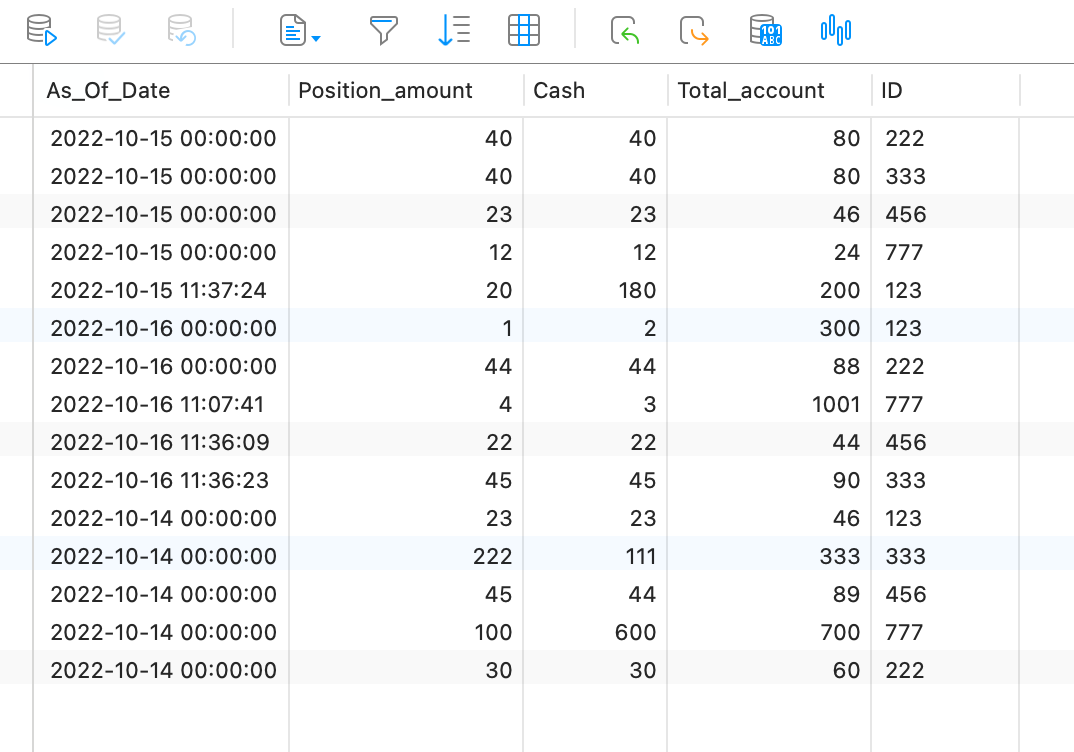
岗位：量化研究员实习

## 笔试说明文档

1. MySQL创建数据库和表



1. 写入样例数据



1. 问题（1）连接数据库

import pymysql

conn = pymysql.Connect(

host = 'bj-cdb-3gfxha84.sql.tencentcdb.com',

port = 59970,

user = 'root',

passwd = 'ainvest\_test1',

db = 'test',

charset = 'utf8'

)

1. 问题（2）画出时间-用户数的走势图

思路：此处时间以天为单位，以时间为横坐标，计算同一天内用户的数量作为纵坐标。

sql = 'SELECT \* FROM account\_balance;'

data = pd.read\_sql(sql, conn)

time\_user\_number\_dict =dict(data.As\_Of\_Date.value\_counts())

time\_user\_number = sorted(time\_user\_number\_dict.items(), key=lambda item:item[0])

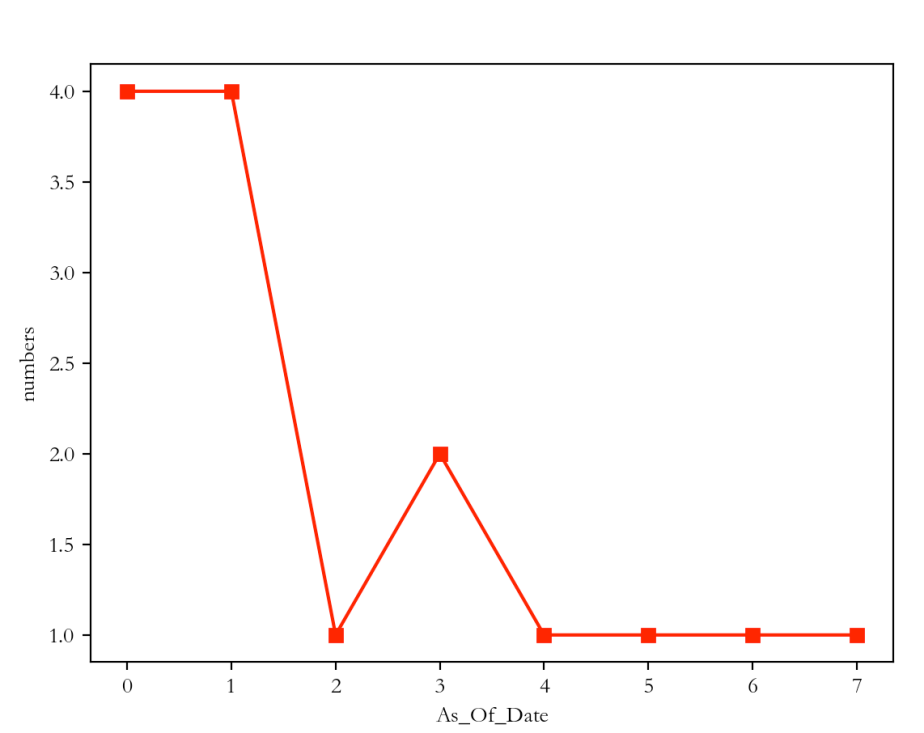
x = range(len(time\_user\_number))

y = list(dict(time\_user\_number).values())

plt.plot(x,y,'s-',color = 'r')

plt.xlabel("As\_Of\_Date")#横坐标名字

plt.ylabel("numbers")#纵坐标名字



1. 问题（3）计算每个用户每天的投资回报率。

使用SQL查询所有时间、ID、总金额，对每个用户遍历，求每个用户的每天的投资回报率。每天的投资回报率，在表中表中计算方式为，先对表以时间排序，并用下一行数据与下一行数据进行计算。最后将每个ID和每天的投资回报率列表，用一个字典来存储起来。

sql = 'SELECT As\_Of\_Date,ID,Total\_account FROM account\_balance;'

data\_history = pd.read\_sql(sql, conn)

user\_list = list(set(data\_history.ID))

user\_return\_rate = {}

for user in user\_list:

user\_data = data\_history[data\_history['ID'] ==user]

# 按时间排序

user\_data.sort\_values(by='As\_Of\_Date')

# 计算投资回报率：(后一行的总金额/前一行的总金额)-1

return\_rate =list((user\_data['Total\_account'].shift(periods=-1, axis=0)/user\_data['Total\_account']).apply(lambda x: x-1))

# 按时间顺序存储每个用户每天的投资回报率

user\_return\_rate[user] = return\_rate[:-1]

输出每个用户的每天的投资回报率如表1 所示

**表1 每个用户每天的投资回报率**

|  |  |  |
| --- | --- | --- |
| ID | 2022-10-15 | 2022-10-16 |
| 456 | -0.4831 | -0.04347 |
| 777 | -0.9657 | 40.7083 |
| 222 | 0.3333 | 0.1000 |
| 333 | 0.125, | 2.7 |
| 123 | 3.34782 | 0.5 |

1. 问题（4）计算每个用户每天的累计回报率

遍历以上字典，计算每个用户的累计回报率

for user,rate\_list in user\_return\_rate.items():

# 累计回报率：每天的回报率+1后，再全部求乘积

reduce\_rate = reduce(lambda x,y:x\*y,[x+1 for x in rate\_list ])

print('用户ID{}的累计回报率为={}'.format(user,reduce\_rate))

输出结果为

用户ID456的累计回报率为=0.49438

用户ID777的累计回报率为=1.42999

用户ID222的累计回报率为=1.46666

用户ID333的累计回报率为=4.16250

用户ID123的累计回报率为=6.52173

1. 问题（5）计算描述性统计

遍历以上的字典，计算每个用户的年华回报率、年华波动率和夏普比率。并存储ID和夏普比率为字典，对字典排序后，输出字典的第一个和最后一个元素，即为夏普比率最大值和最小值。

# 年化回报率

sharpe\_ratio\_dict = {}

for user,rate\_list in user\_return\_rate.items():

# 年化回报率

year\_return\_rate = np.mean(rate\_list)\*252

# 年化波动率

year\_volatility = np.std(rate\_list,ddof=1)\* np.sqrt(252)

# 夏普比率

sharpe\_ratio = year\_return\_rate/year\_volatility

sharpe\_ratio\_dict[user] = sharpe\_ratio

# 输出夏普比率最高和最低的用户

sharpe\_ratio\_dict\_sorted= dict(sorted(sharpe\_ratio\_dict.items(), key=lambda item:item[1]))

print('夏普比率最低的用户是{},夏普比率为{}'.format(sharpe\_ratio\_dict\_sorted.keys()[0],sharpe\_ratio\_dict\_sorted.values()[0]))

print('夏普比率最高的用户是{},夏普比率为{}'.format(sharpe\_ratio\_dict\_sorted.keys()[-1],sharpe\_ratio\_dict\_sorted.values()[-1]))

输出结果为

夏普比率最低的用户是456,夏普比率为-13.44502

夏普比率最高的用户是222,夏普比率为20.84637

8、问题（6）输出前95%的最后一天日期和后5%最后一天日期

# 输出前95%的最后一天日期和后5%最后一天日期。

user\_data = data\_history[data\_history['ID'] ==user\_list[0]]

# 按时间排序

user\_data.sort\_values(by='As\_Of\_Date')

# 前95%的数据跨度天数

data\_len = int(len(rate\_list)\*0.95)

print('前95%的最后一天日期是{}'.format(user\_data['As\_Of\_Date'].iloc[data\_len-1]))

print('后5%的最后一天日期是{}'.format(user\_data['As\_Of\_Date'].iloc[len(user\_data)-1]))

输出结果为

前95%的最后一天日期是2022-10-15 00:00:00

后5%的最后一天日期是2022-10-16 15:38:57

1. 问题（6）简单线性回归

X = []

Y = []

for user,rate\_list in user\_return\_rate.items():

# 第一步计算样本前95%的年化回报率

x\_year\_return\_rate = np.mean(rate\_list[:data\_len])\*252

# 第二步计算样本后5%的年化回报率

y\_year\_return\_rate = np.mean(rate\_list[data\_len+1:])\*252

X.append(x\_year\_return\_rate)

Y.append(y\_year\_return\_rate)

# 第三步，线性回归分析

X = sm.add\_constant(X) # 若模型中有截距，必须有这一步

model = sm.OLS(Y, X).fit() # 构建最小二乘模型并拟合

print(model.summary()) # 输出回归结果

由于给出的样本量太少，此处举例说明一元回归分析过程

import statsmodels.api as sm

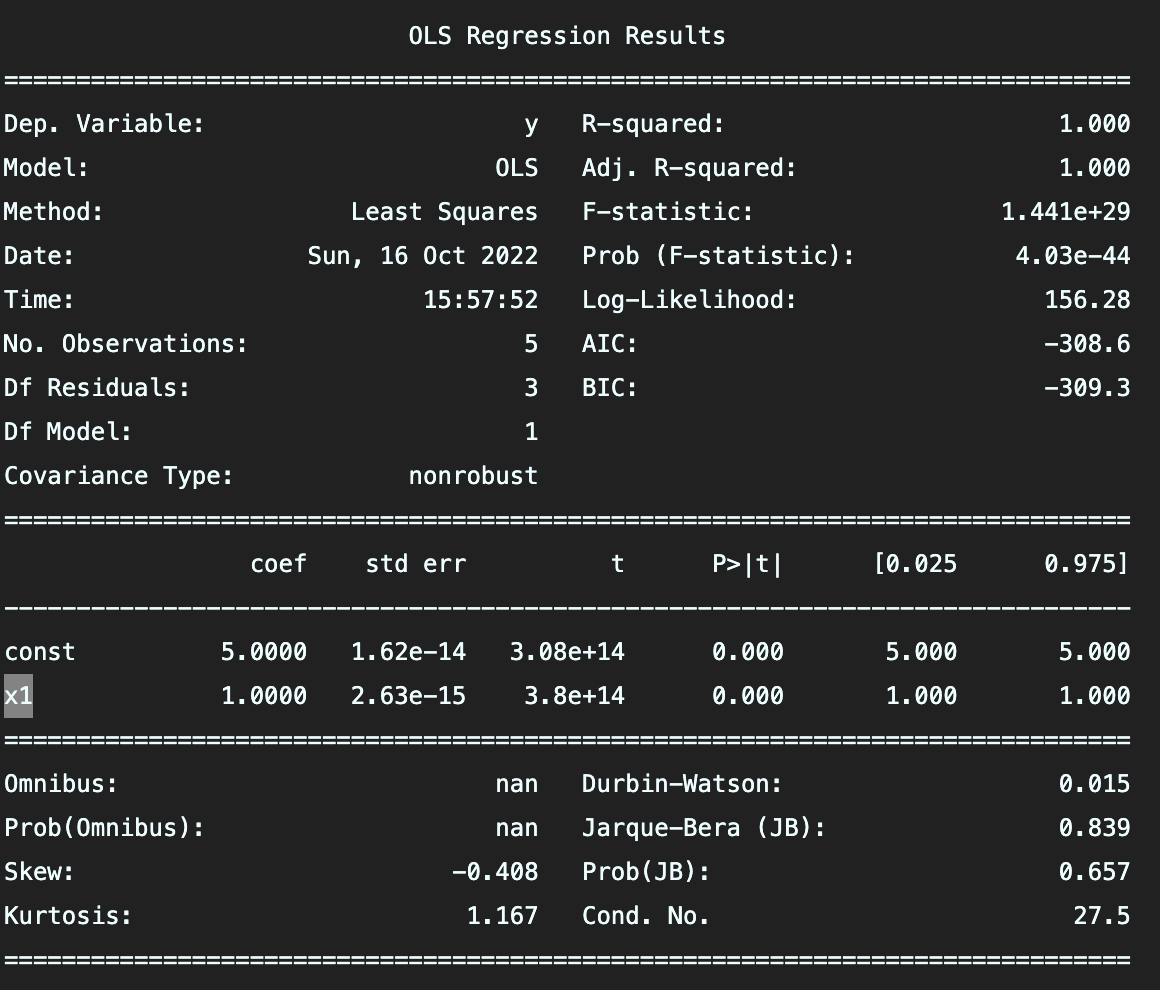
X = [4,5,6,7,8]

Y = [9,10,11,12,13]

X = sm.add\_constant(X) # 若模型中有截距，必须有这一步

model = sm.OLS(Y, X).fit() # 构建最小二乘模型并拟合

print(model.summary()) # 输出回归结果



根据打印出的回归分析结果，知道一元回归的系数x1 = 1,大于0，则是正相关，常数b = 5，即一元回归方程为

y = x1+5

其中R平方为决定系数，反应因变量的全部变异能通过回归关系被自变量解释的比例，一般机器默认的是r2>0.99，这样才具有可行度和线性关系。在分析结果中未1，说明具有线性关系。

其中std err代表了实际值和预计值的偏差的平方和，很明显std err越大，偏差越大。因此，一个比较小的std err，反映了一个更好的预测结果。在分析结果中,std err接近0，说明具有预测效果较佳。