台北富邦銀行 Taipei Fubon Bank

鷹眼識詐聯盟技術交流

數據科學部 劉晉儒 2023.11.24

資料欄位問題

No	欄位名稱	問題	回答
1	ACCT_NBR_ORI母帳號	1.貴行母帳號之定義？ 2.本行未區分母、子帳號，可否用同一個帳號建檔？ 3.或帳號是指母帳號底下之虛擬帳號？	母帳號為自然人台外幣活存綜合、數位存款帳號，為存摺帳號。 例:一個人有兩本存摺則有兩個母帳號。如無母帳號架構，則以存摺帳號即可。子帳號依功能別區分，類似虛擬帳號。
2	AUM(NTD, 萬元)	1. 資料要台幣和外幣，還是只要台幣？ 2. 客戶基本資料檔AUM_AMT欄位，定義為何？	台外幣存款支存+存款定存+存款活存+信託其他+信託指單 +海外股票+國外基金+國內基金+國內基金代銷+組合式商品+ 債券+債券附條件+黃金存摺，為現在市值折台幣庫存餘額
3	CHANNEL_DESC通路說明	1.通路說明跟通路CHAL_1之差別？ 2.通路說明及通路的種類，貴行各有哪些？ 3.因本欄位無字元限制，可否用代號表示，如網路銀行用1 及貴行是否有相關之代號設定？ 4.通路說明，通路說明與編號30的通路，只有合併的一個欄位，有影響嗎？	與通路CHAL_1不同，CHAL_1區分為ATM、Online、 Mobile、Branch、Batch，無須與富邦代碼對齊，有以下 分類即可:網銀、行銀、自行ATM、跨行ATM、銀資交換匯 款、一卡通、端末(臨櫃)
4	EMP_NO櫃員編號	1.本行目前櫃員卡兩碼，需要補滿嗎 2.櫃員編號，目前無法對應回員編，會影響後續分析嗎？	富邦代碼如下(01、02、03為不同路徑)，需有以下分類，無 須與富邦代碼對齊 4001:跨行提款 4002:跨行轉帳 4005:跨行匯入 4011:網路銀行櫃員01 4012:網路銀行櫃員02 4013:網路銀行櫃員03 4014:行動銀行 4021:晶片卡跨行提款 4022:晶片卡跨行轉帳 4111:行動銀行櫃員01 4112:行動銀行櫃員02 4115:網路銀行櫃員 4215:新行動銀行 3開頭:自動櫃員機

資料欄位問題

No	欄位名稱	問題	回答
5	LOGIN_LOG_KEY登入記錄鍵值	1.登入記錄鍵值之定義？ 2是否可用KEY值，或本欄要求9碼或有特殊格式？	LOGIN_LOG_KEY為登入記錄鍵值，為該次登入紀錄的鍵值。可自行以編碼。
6	MENU_ID網銀選單代碼	1. 資料要台幣和外幣，還是只要台幣？ 2. 客戶基本資料檔AUM_AMT欄位，定義為何？	行銀:SQL可改為自行的代碼，主要為查餘額及調整額度相關 MDS01、MDS0101:存款-->我的存款 MDS04、MDS0401、MDS0402:存款/外匯/轉帳-->交易查詢 MPS1302:個人服務-->提高轉帳額度。 網銀:SQL可改為自行的代碼，主要為查餘額及調整額度相關 CDS04、CDS0401:存款-->我的存款 CDS01、CDS0102:存款/外匯/轉帳-->我的存款 CBO03、CDF02、CDF04:資產負債-->我的存款、存款交易查詢
7	ALERT_FLG警示戶	1.警示戶(ALERT_FLG)是指彙整完所有的帳號後，再根據是否為警示戶押上Y/N的註記嗎? 2.這個註記要by帳號所屬人名下的所有帳號還是by個別帳號標記?	指法院、檢察署或司法警察機關為偵辦刑事案件需要，通報設為警示帳戶之存款帳戶。 提供警示帳戶帳號並標紀警示帳戶的資料即可。

AGENDA

01 資料處理流程

02 現有排程架構

03 資料清理

04 特徵工程前置作業

資料處理流程

資料處理流程

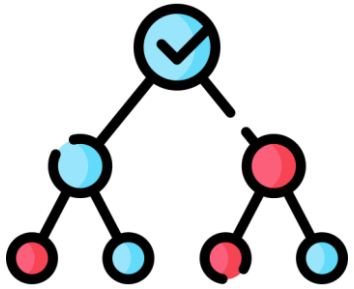
資料蒐集

- 定義問題及目標
- 資料源選擇
- SQL蒐集資料

資料前處理

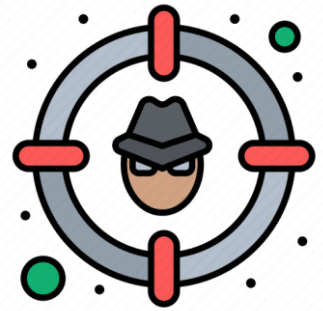
- 資料整合
- 資料清理
- 特徵工程

建置模型



- 切分訓練/測試資料
- 建置模型
- 調整參數

模型預測



- 準備資料
- 建立排程預測
- 產出可疑名單
- 成效追蹤

資料處理流程

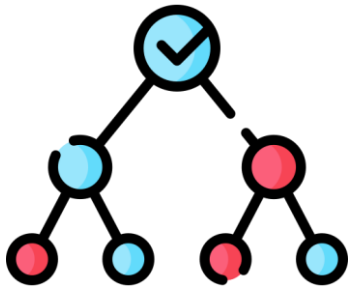
資料蒐集

- 定義問題及目標
- 資料源選擇
- SQL蒐集資料

資料前處理

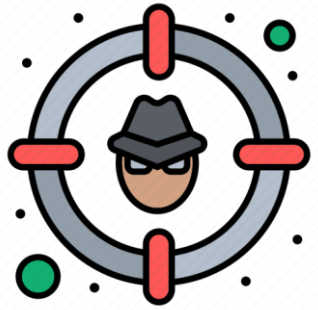
- 資料整合
- 資料清理
- 特徵工程

建置模型



- 切分訓練/測試資料
- 建置模型
- 調整參數

模型預測



- 準備資料
- 建立排程預測
- 產出可疑名單
- 成效追蹤

1st 技術交流

2nd 技術交流

2nd 技術交流

現有排程架構

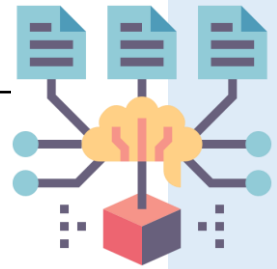
現有排程架構(Model Serving)

現有排程架構(Model Serving)

程式碼

Job_SQL語法

POLICE_DASB_job_cust_pred.sql
POLICE_DASB_salary_acct.sql
POLICE_DASB_job_tracking_log_pred.sql

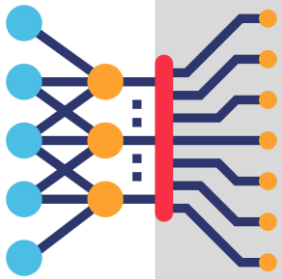


Job 排程腳本

01.data_cleaning.py
02.feature_engineering_1.py
03.feature_engineering_2.py

Utils函式

config_tbl.py
data_cleansing.py
feature_engineering_2023.py
model_testing.py



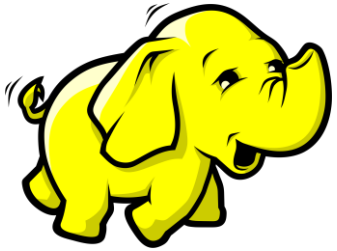
Model模型

xgb_model_2022-08-31_fin_v1
second_xgb_model_2022-08-31_fin_v1

資料清理

資料準備及清理

準備蒐集



Hadoop

薪轉戶資料

Oracle SQL

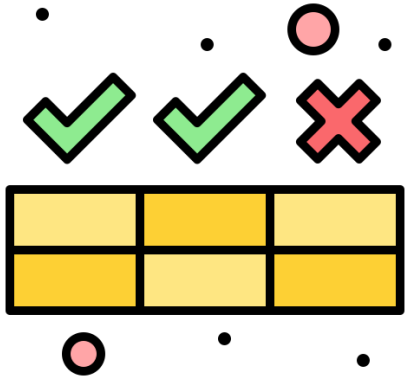
警示戶資料
證券戶資料

各銀行略有差異，
請自行機動調整

排除特定帳戶

排除薪轉、警示、證券戶
及特定職業

資料清理



保留特定通路、交易類型，
並去除特定交易、換匯交易

資料蒐集

01.警示戶資料 → 02.證券戶資料 → 03.薪轉戶資料 → 04.保留檢視帳戶

修改utils: config_tbl.py

```
# 我整支程式只有一個參數檔
HIVE_VIEW = 'usr_XXX_XX'
ORACLE_VIEW = 'usr_XXX_XX'
BANCS_CUST_ACCT_EXP = 'BANCS_CUST_ACCT_EXP'          核心客戶檔 (已上警示之帳戶)
SECURITY = 'ASHF'                                      證券戶檔

SAV_TXN_VIEW = 'DM_T_VIEW'
SAV_TXN = 'WMG_SAV_TXN'                                活存交易檔
WMG_CUST_VIEW = 'DM_T_VIEW'
WMG_CUST = 'wmg_cust'                                客戶基本資料檔

MB_LOG_VIEW = 'MB_LOG_VIEW'
MB_ACCESS_LOG = 'MB_ACCESS_LOG'                        行銀軌跡檔
B2C_LOG_VIEW = 'B2C_LOG_VIEW'
B2C_ACCESS_LOG = 'B2C_ACCESS_LOG'                      網銀軌跡檔
```

資料蒐集

01.警示戶資料 → 02.證券戶資料 → 03.薪轉戶資料 → 04.保留檢視帳戶

Job: 01.data_cleaning.py

```
##################
### 核心客戶檔 ###
##################
query1 = f'''
(
SELECT ACCT_NO_SAV, DERIVATIVE_DATE, DERIVATIVE_FLG, SWINDLE_MORATORIUM_DATE, SWINDLE_MORATORIUM_FLG, SWINDLE_FLG, ALERT_DATE, ALERT_FLG
FROM {ORACLE_VIEW}.{BANCS_CUST_ACCT_EXP}
WHERE DERIVATIVE_FLG is not null
OR SWINDLE_MORATORIUM_FLG is not null
OR SWINDLE_FLG is not null
OR ALERT_FLG is not null
) cctlog
'''
```

#修改表名

資料蒐集

01.警示戶資料 → 02.證券戶資料 → 03.薪轉戶資料 → 04.排除左列帳戶

Job: 01.data_cleaning.py

```
##################
### 撈取證券戶 ###
##################
# TODO_8: 修改證券客戶XX: 證券商及保代號

query2 = f'''
(select distinct DEP_ACCT_NO
from {ORACLE_VIEW}.{SECURITY} #修改表名
where (substr(STCK_CSTDY_CD,1,2) = 'XX'
OR STCK_CSTDY_CD is null
OR STCK_CSTDY_CD = 'XX'
OR length(STCK_CSTDY_CD) < 4))cctlog
'''
```

資料蒐集

01.警示戶資料 → 02.證券戶資料 → 03.薪轉戶資料 → 04.排除左列帳戶

Job: 01.data_cleaning.py

由impala撈取薪轉戶資料

```
########################################
#### 更改當下路徑到運行腳本的資料夾路徑 ####
########################################

# 將工作路徑改到此 script 的路徑
abspath = os.getcwd()
print(f'Original absolute path: {abspath}')
print('Change working directory...')
os.chdir(r'/home/cdsw/sql')  # 因為之後要用到自己寫的 function，所以一定要改目錄到相應的位置
abspath = os.getcwd()
print(f'Current absolute path: {abspath}')

#建立薪轉戶資料表
tbl_name = 'salary_acct'

try:
    # Create impala table
    impala_sql(f'POLICE_DASB_{tbl_name}.sql', logger, mode='overwrite', table_name = f'usr_julian_liu.POLICE_DASB_salary_acct_temp')
    # impala to Hive
    query = f'''
    SELECT *
    FROM usr_julian_liu.POLICE_DASB_salary_acct_temp
    '''
    time.sleep(10)
    sdf_salary = spark.sql(query)
    sdf_salary.show(2)
    # Create Hive formal table
    sdf_salary.write.format('Hive').mode('overwrite').saveAsTable(f'usr_julian_liu.POLICE_DASB_salary_acct')
except:
    print('Salary Account Table Creation failed.')
```

資料蒐集

01.警示戶資料 → 02.證券戶資料 → 03.薪轉戶資料 → 04.排除左列帳戶

Job_sql: POLICE_DASB_salary_acct.sql

```
SELECT distinct(acct_nbr_ori) acct_no_14,
flg_pr payroll_code
FROM (
    SELECT distinct A.acct_nbr_ori, A.cust_id, B.flg_pr
    FROM SAV_TXN_VIEW.SAV_TXN A
    INNER JOIN (
        SELECT cust_id, flg_pr
        FROM WMG_CUST_VIEW.WMG_CUST
        WHERE flg_pr = 'Y'
    ) B
    ON A.cust_id = B.cust_id
    WHERE flg_pr = 'Y'
) T1
```

資料蒐集

01.警示戶資料 → 02.證券戶資料 → 03.薪轉戶資料 → 04.排除左列帳戶

Job: 01.data_cleaning.py

清除已警示、薪轉戶、證券戶資料

```
####################################
########## 修改sql時間區間 ##########
####################################
tbl_name = 'job_cust_predict'
# open file
with open(f'POLICE_DASB_{tbl_name}.sql',"r") as f:
    string = f.read().replace('SNAP_DATE',str(max_date))

# write as tmp.file
with open(f'POLICE_DASB_{tbl_name}_tmp_file.sql',"w") as f:
    f.write(string)

#####################################
#### 由impala撈取排除條件之預測資料 ####
#####################################
try:
    #建立normal_acct帳號資料
    tbl_name = 'job_cust_predict'
    # Create impala table
    impala_sql(f'POLICE_DASB_{tbl_name}_tmp_file.sql', logger, mode='overwrite', table_name = f'usr_julian_liu.POLICE_DASB_{tbl_name}_tmp')
    time.sleep(10)
    print('Normal_acct Table Creation done!')
except Exception:
    logging.exception('message')
    print('Normal_acct Table Creation failed.')
```

資料蒐集

01.警示戶資料 → 02.證券戶資料 → 03.薪轉戶資料 → 04.排除左列帳戶

Job: 01.data_cleaning.py

```
SELECT acct_nbr_ori, acct_nbr, act_date, tx_date, tx_time, drcr, cust_id, own_trans_acct, tx_amt, pb_bal, tx_brh, cur, channel_desc, chal_1, memo, remk, jrnl_no, tx_mode
FROM [VIEW].[TABLE NAME]                                    --請修改活存交易檔[VIEW].[TABLE NAME] (VIEW.活存交易檔名ex.WMG_SAV_TXN)
WHERE acct_nbr_ori not in (
        /*排除所有警示戶、薪轉戶、證券戶*/
        SELECT distinct acct_nbr_ori AS acct_nbr_ori
        FROM (
            /*已為警示戶*/
            SELECT acct_no_sav AS acct_nbr_ori
            FROM HIVE_VIEW.BANCS_CUST_ACCT_EXP
            WHERE alert_flg = 'Y'
            UNION ALL
            /*排除薪轉*/
            SELECT distinct acct_no_14 AS acct_nbr_ori
            FROM HIVE_VIEW.POLICE_DASB_job_salary_acct
            UNION ALL
            /*排除證券*/
            SELECT distinct dep_acct_no AS acct_nbr_ori
            FROM HIVE_VIEW.POLICE_DASB_ashf_list
        ) T
)
AND cust_id not in (
    /*排除會計師醫師律師*/
    SELECT DISTINCT CUST_ID AS cust_id
    FROM HIVE_VIEW.WMG_CUST                                 --請修改客戶基本資料檔[VIEW].[TABLE NAME]
    WHERE (OCCUPATION IN ('20', '25', '26') OR AGE <= 10)
    AND LENGTH(CUST_ID) = 10
    AND LEFT(CUST_ID, 1) rlike '[a-zA-Z]'
)
AND length(cust_id) = 10
AND LEFT(CUST_ID, 1) rlike '[a-zA-Z]'
AND TX_MODE = '1'
AND (substr(emp_no, 4, 4) in ('4001','4002','4005','4011','4012','4013','4014','4021','4022','4111','4112','4115','4215')
OR substr(emp_no, 4, 1) = '3')
AND tx_date BETWEEN ADD_MONTHS('SNAP_DATE', -1) AND ('SNAP_DATE')
ORDER BY acct_nbr_ori ASC, tx_date ASC, tx_time ASC
```

資料清理

保留通路

編號	交易通路內容
1	銀行資訊交換平台FEP跨行ATM
2	網路銀行 行銀
3	銀行資訊交換平台FEP匯款
4	銀行資訊交換平台FEP自行ATM
5	新端末
6	網路銀行 網銀
7	一卡通

保留摘要

編號	摘要內容
1	CD轉收
2	CD提款
3	行銀跨轉
4	匯入款
5	行動自轉
6	網路跨轉
7	CD轉支

清理備註

編號	備註內容
1	信用卡
2	學貸
3	房貸
4	證券
5	基金
6	信貸
7	保險

資料清理

01.py檔流程 → 02.清除正常交易交易 → 03.清除換匯交易

Job: 01.data_cleaning.py

```
################################
######## 正常戶交易清理 ########
################################
try:
    #add logging
    logger.info(f"{'=' * 20} Normal Acct Transaction Cleansing started {'=' * 20}")
    start = time.time()
    tbl_name = 'job_cust_predict'
    query02 = f'''
    SELECT *
    FROM usr_julian_liu.POLICE_DASB_{tbl_name}_tmp
    '''
    #==================原cleansing函數==================
    nonwarning = spark.sql(query02)
    data = nonwarning
    data = data.cache()
    data = data.withColumn('drcr', data.drcr.cast('integer'))
    data = data.withColumn('pb_bal', data.pb_bal.cast('float'))
    data = data.withColumn('tx_amt', data.tx_amt.cast('float'))
    data = data.withColumn('tx_date', date_format(data.tx_date,'yyyy-MM-dd'))
    data = data.withColumn('tx_time', to_timestamp(concat(data.tx_date, data.tx_time),'yyyy-MM-ddHHmmss'))
    data = dc.salary_list_cleansing(data, spark)
    data = data.filter(data['tx_mode'] == 1)
    data = dc.channel_memo_cleansing_func(data)
    data = dc.exchange_cleansing_func(data) #排除換匯交易
    data = dc.filter_cur(data) #排除其他純外幣交易
    #===================================
```

#清理薪轉戶(可忽略)
#保留正常交易
#清理摘要
#排除換匯交易
#排除純外幣交易

資料清理

01.py檔流程 → 02.清除正常交易交易 → 03.清除換匯交易

utils: data_cleansing.py

```
def channel_memo_cleansing_func(data: F.DataFrame):
    """
    功能: 限定交易通路/memo/排除字串備註
    input: 警示戶交易資料Spark Dataframe
    output: 警示戶交易資料Spark Dataframe
    """
    print('channel_memo cleansing')
    print('='*70)
    channel_desc = ['銀行資訊交換平台FEP跨行ATM','網路銀行 行銀','銀行資訊交換平台FEP匯款',
                    '銀行資訊交換平台FEP自行ATM','新端末','網路銀行 網銀','一卡通']

    tx_list = \
     ['ＣＤ轉收','ＣＤ提款','行動跨轉','匯入款','行動自轉','網路跨轉','ＣＤ轉支','跨行存款',
    'ＣＤ存現','領現','存現','轉支','提領','主動儲值','轉收','ＣＤ轉帳', '儲值',
    '跨國提款','網路自轉','行動換匯','無卡提款','付款','國外匯入','國外匯出']

    remk_remove=['信用卡|信用|卡費|學貸|房|房貸|房租|租|關稅|稅|費|租金|訂|訂金|停車|車|住宿|機票|理賠|信貸|基金|保險|期貨|證券|\
                 健保|醫療|診所|勞保|薪資|薪轉|水費|電費|水電|瓦斯|電信|年金|津貼|學雜費|報名|\
                 學費|利息|育嬰|育兒|補習|課補|刷卡|人壽|奈米投|代繳|獎金|帳單|罰單|急用|繳|股份|有限|紅包|款|科技|電']
    data = data[data.channel_desc.isin(channel_desc)]
    data = data[data.memo.isin(tx_list)]
    data = data.filter(~data.remk.contains('|'.join(remk_remove)))
    return data
```

#自行依據交易內容調整排除條件

資料清理

01.py檔流程 → 02.清除正常交易交易 → 03.清除換匯交易

utils: data_cleansing.py

```
def exchange_cleansing_func(data: F.DataFrame):
    """
    功能: 排除換匯交易
    input: 警示戶交易資料Spark Dataframe
    output: 警示戶交易資料Spark Dataframe
    步驟:
    step1: 新增單調遞增index作為每列編號
    step2: 以母帳號、交易時間為group，計算同一時間交易次數
    step3: 篩選同一時間交易筆數>2筆資料
    step4: 計算同一帳號同一時間交易同一交易序號出現多少次不同的幣別
    step5: 篩選不同的幣別
    step6: 計算同一帳號同一時間交易同一交易序號出現多少次不同的轉入轉出
    step7: 篩選轉出又轉入的交易
    step8: 篩選不同的幣別的表並mapping轉出又轉入的交易
    step9: 篩選換匯交易之index
    step10: 換匯交易刪除
    """
    print('exchange_cleansing')
    print('='*70)

    data = data.withColumn('exchange_remove_index', row_number().over(Window.orderBy(monotonically_increasing_id())))
    tx_time_dupli = data.groupBy('acct_nbr_ori','tx_time').agg(F.count('tx_time').alias('count'))
    tx_time_dupli = tx_time_dupli.filter(tx_time_dupli['count']>2)
    tx_count = data.join(tx_time_dupli, ['acct_nbr_ori', 'tx_time'])
    cur_exchange = tx_count.groupBy(['acct_nbr_ori','tx_time','jrnl_no']).agg(countDistinct('cur').alias('count_cur'))
    cur_exchange = cur_exchange.filter(cur_exchange['count_cur']>1)
    drcr_exchange = tx_count.groupBy(['acct_nbr_ori','tx_time','jrnl_no']).agg(countDistinct('drcr').alias('count_drcr'))
    drcr_exchange = drcr_exchange.filter(drcr_exchange['count_drcr']>1)
    exchange = cur_exchange.join(drcr_exchange, on=['acct_nbr_ori','tx_time','jrnl_no'])
    tx_count = tx_count.join(exchange, on=['acct_nbr_ori','tx_time','jrnl_no'])
    tx_count = tx_count.select('exchange_remove_index')
    tx_count = tx_count.withColumn('idx', lit(1))
    data = data.join(tx_count, on=['exchange_remove_index'], how='left')
    data = data.filter(data.idx.isNull()).drop('exchange_remove_index','idx')
    return data
```

特徵工程前置作業

特徵工程前置作業

網行銀軌跡資料

ID	access_date	Login_log_key	Menu_id	Error_desc
A12345678*	2023-05-01	3477˙970*	MDS01	0000
A12345678*	2023-05-02	3477˙970*	MDS04	0000
A12345678*	2023-05-02	3477˙970*	MPS1302	0000

ID	access_date	Login_log_key	Menu_id	Error_desc
A12345678*	2023-05-01	3777˙980*	CDS04	0000
A12345678*	2023-05-02	3777˙980*	CDS03	0000

特徵工程前置作業

計算餘額查詢次數

Job_sql: POLICE_DASB_job_tracking_log_pred.sql

```sql
SELECT company_uid, access_date, SUM(CASE WHEN category = 'MB_limit' THEN times END) MB_limit, SUM(CASE WHEN category = 'MB_check' THEN times END
FROM(
    SELECT company_uid, access_date, category, COUNT(DISTINCT login_log_key) as times
    FROM (
        SELECT
            company_uid,
            login_log_key,
            to_date(access_time) AS access_date,
            error_desc,
            menu_id,
            CASE
                WHEN menu_id IN ('MDS01', 'MDS0101', 'MDS04', 'MDS0401', 'MDS0402') THEN 'MB_check' --查餘額
                WHEN menu_id IN ('MPS1302') THEN 'MB_limit' --行銀調額
            END AS category
        FROM ODS_T_VIEW.MB_ACCESS_LOG
        WHERE company_uid IN (
            SELECT distinct cust_id
            FROM usr_julian_liu.job_cust_id_pred
        )
        AND menu_id in ('MDS01', 'MDS0101', 'MDS04', 'MDS0401', 'MDS0402','MPS1302')
        AND from_unixtime(unix_timestamp(access_date), 'yyyy-MM-dd') BETWEEN (SELECT min(min_date) FROM usr_julian_liu.job_cust_id_pred)
        AND (SELECT max(max_date) min_date FROM usr_julian_liu.job_cust_id_pred)
        UNION ALL
        SELECT
            company_uid,
            login_log_key,
            to_date(access_time) AS access_date,
            error_desc,
            menu_id,
            CASE
                WHEN menu_id in ('CDS0401', 'CDS04', 'CDS01', 'CDS0102', 'CBO03', 'CDF02', 'CDF04') THEN 'EB_check' --查餘額
            END as category
        FROM ODS_T_VIEW.B2C_ACCESS_LOG
        WHERE company_uid IN (
            SELECT distinct cust_id
            FROM usr_julian_liu.job_cust_id_pred
        )
        AND menu_id in ('CDS0401', 'CDS04', 'CDS01', 'CDS0102', 'CBO03', 'CDF02', 'CDF04')
        AND from_unixtime(unix_timestamp(access_date), 'yyyy-MM-dd') BETWEEN (SELECT min(min_date) FROM usr_julian_liu.job_cust_id_pred)
        AND (SELECT max(max_date) min_date FROM usr_julian_liu.job_cust_id_pred)
    ) t
    GROUP BY company_uid, access_date, category
) p
GROUP BY company_uid, access_date
```

ID	access_date	category
A123456789	2023-05-01	MB_check
A123456789	2023-05-02	MB_check
A123456789	2023-05-02	MB_limit
A123456789	2023-05-04	MB_check

ID	access_date	MB_check	MB_limit
A123456789	2023-05-01	1	0
A123456789	2023-05-02	1	1
A123456789	2023-05-04	1	0

Q&A


Taipei Fubon Bank
Thank You!