Stock 经销商数据

1. 项目背景

1.1. 简介

该项目整体主要围绕bosch的合作伙伴——经销商的库存数据，其目的是为了将所有经销商的数据合并作为中台的数据资产，为销售的PBI展示提供数据支撑，为公司的运营决策提供数据引导。

> - bosch的经销商数据会不时增加、上传数据时期也各不相同
> - Bosch是一家德国跨国工业集团，总部位于斯图加特，在全球范围内拥有超过440个子公司和地区公司，业务覆盖领域包括汽车、家用电器、能源技术、工业技术和建筑技术等。该公司成立于886年，创始人为罗伯特·博世（Robert Bosch），目前全球拥有约40万名员工，年销售额超过730亿欧元。Bosch以其优秀的工程技术和革新性产品闻名于世界，是业界领先的汽车零部件制造商和供应商之一。同时，Bosch还致力于推动智能化技术和物联网技术的发展，在智能家居、工业4.0等领域也有着较为广泛的应用和创新成果。

1.2. 需求

1.2.1. 料号匹配表

- 部分经销商的料号有他们自己的bosch料号转换方式，需要将这些表接进中台，目前还未正式上线

Target Table			Source Tables				Transformation
Column Name	Data Type	Primary Key	System	Table Name	Column Name	Data Type	
product_code	String	PK	NFS（文档服务器）-> blob storage	dealer_product_code_mapping_118xxxx	product_code	String	取blob storage中csv文件，不同的经销商会提供多个product code mapping文件，上传NFS后通过Redlake satellite上传到blob storage再抽取到数据中台 测试阶段使用路径 bosch-data-warehouse / test11下的csv文件dealer_product_code_mapping_118002298.csv 上线需切换到正式路径
boschpartno	String		NFS（文档服务器）-> blob storage		boschpartno	String	

表名：ods_dealer_product_code_mapping

1.2.2. 料号整合表

- 由将各个经销商的的料号匹配表数据合并

Target Table			Source Tables				Transformation
Column Name	Data Type	Primary Key	System	Table Name	Column Name	Data Type	
customercode	String	PK	datasimba	ods_dealer_product_code_mapping_118xxxx			取每张dealer_product_code_mapping的ods表名中118xxxxx 经销商号
product_code	String	PK	datasimba	ods_dealer_product_code_mapping_118xxxx	product_code	String	
boschpartno	String		datasimba	ods_dealer_product_code_mapping_118xxxx	boschpartno	String	

表名：dwd_del_dealer_product_code_map

1.2.3. 经销商数据增量整合表

- 将ods自动采集的各个经销商的最新一天数据进行合并，处理转换，形成一张增量表

Target Table			Source Tables				Transformation
Column Name	Data Type	Primary Key	System	Table Name	Column Name	Data Type	
boschpartno13	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df	boschpartno13	String	
productname	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df	productname	String	
productcategory	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df	productcategory	String	
stockqty	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df	stockqty	String	
unit	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df	unit	String	
loaddate	String	PK	Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df	loaddate	String	统一格式为YYYYMMDD,只取最近一天得库存数据
loadtime	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df	loadtime	String	统一格式为HH:MM:SS
boschpartno_verified	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df ods_azure_blob_ymtk10001_cross_ref_info_df ods_azure_blob_v_ymtk00101_df ods_azure_blob_v_aamm_gen_material_df ods_spiderb_sys_product_df		String	根据ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df表的boschpartno字段，按以下步骤进行清洗 1、和【dwd_del_dealer_product_code_mapping】中的【product_code】进行匹配，成功则输出此表中的【boschpartno】 2、清洗掉除字母、数字之外的符号后取前10位 3、和“gen_material表”中的“10位料号”进行匹配——ods_azure_blob_v_aamm_gen_material_df【material_10_digits】,如匹配上则填入【material_10_digits】，匹配不上进入步骤3 4、和 【ods_azure_blob_v_ymtk00101_df 】中的 “alternative number”【 comp_alt_no】 进行匹配，匹配成功输出此表中的“物料”字段【material】 5、和 【ods_azure_blob_ymtk10001_cross_ref_info_df】中的“原厂料号”【khnr_verd】进行匹配，成功则输出此表中“料号”字段【matnr】 6、如以上步骤都不能匹配到正确的博世料号，则输出0

表名：dwd_latest_dealer_stock

1.2.4. 经销商历史数据全量整合表

- 包含整合处理过后的所有经销商的历史数据，每日增量更新

Target Table			Source Tables				Transformation
Column Name	Data Type	Primary Key	System	Table Name	Column Name	Data Type	
customercode	String	PK	Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	customercode	String	
warehousename	String	PK	Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	warehousename	String	
warehouseno	String		Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	warehouseno	String	
boschpartno	String	PK	Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	boschpartno	String	
boschpartno13	String		Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	boschpartno13	String	
productname	String		Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	productname	String	
productcategory	String		Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	productcategory	String	
stockqty	String		Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	stockqty	String	
unit	String		Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	unit	String	
loaddate	String	PK	Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	loaddate	String	统一格式为YYYYMMDD，取历史上进销商传输过来得每天得库存数据
loadtime	String		Datasimba	第一次取blob中历史文件得全量采集，以后将dwd_latest_dealer_stock增量合并到本表得全量数据中	loadtime	String	统一格式为HH:MM:SS
boschpartno_verified	String		Datasimba	ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df ods_azure_blob_ymtk10001_cross_ref_info_df ods_azure_blob_v_ymtk00101_df ods_azure_blob_v_aamm_gen_material_df ods_spiderb_sys_product_df		String	根据ods_azure_blob_auto_dealer_stock_118xxxxxxxx_df表的boschpartno字段，按以下步骤进行清洗 1、和【dwd_del_dealer_product_code_mapping】中的【product_code】进行匹配，成功则输出此表中的【boschpartno】 2、清洗掉除字母、数字之外的符号后取前10位 3、和“gen_material表”中的“10位料号”进行匹配——ods_azure_blob_v_aamm_gen_material_df【material_10_digits】,如匹配上则填入【material_10_digits】，匹配不上进入步骤3 4、和 【ods_azure_blob_v_ymtk00101_df 】中的 “alternative number”【 comp_alt_no】 进行匹配，匹配成功输出此表中的“物料”字段【material】 5、和 【ods_azure_blob_ymtk10001_cross_ref_info_df】中的“原厂料号”【khnr_verd】进行匹配，成功则输出此表中“料号”字段【matnr】 6、如以上步骤都不能匹配到正确的博世料号，则输出0

表名：dwd_dealer_stock_history

1.2.5. 经销商库存月览表

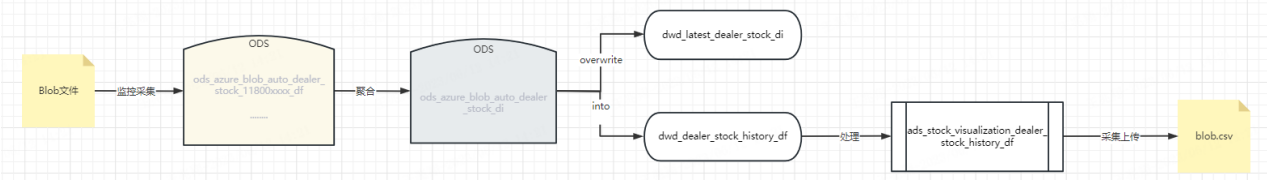
- 包含当月的库存数据和其他月份月初月末的库存数据

Target Table			Source Tables				Transformation
Column Name	Data Type	Primary Key	System	Table Name	Column Name	Data Type	
customercode	String	PK	Datasimba	dwd_dealer_stock_history		String	
warehousename	String	PK	Datasimba	dwd_dealer_stock_history	warehousename	String	
warehouseno	String		Datasimba	dwd_dealer_stock_history	warehouseno	String	
boschpartno	String	PK	Datasimba	dwd_dealer_stock_history	boschpartno	String	
boschpartno13	String		Datasimba	dwd_dealer_stock_history	boschpartno13	String	
productname	String		Datasimba	dwd_dealer_stock_history	productname	String	
productcategory	String		Datasimba	dwd_dealer_stock_history	productcategory	String	
stockqty	String		Datasimba	dwd_dealer_stock_history	stockqty	String	
unit	String		Datasimba	dwd_dealer_stock_history	unit	String	
loaddate	String	PK	Datasimba	dwd_dealer_stock_history	loaddate	String	统一格式为YYYYMMDD，取当前月份每天得数据，以及过去月份每月月初和月末得数据
loadtime	String		Datasimba	dwd_dealer_stock_history	loadtime	String	
boschpartno_verified	String		Datasimba	dwd_dealer_stock_history		String	

表名：ads_stock_visualization_stock

2. 项目架构

2.1. 流程图



2.2. 数据层级

- Blob：数据文件源，由各个经销商上传json或者parquet文件
- ODS：临时数据存储，用来存储从Blob中自动采集生成得数据表
- DWD：数据处理，匹配其他维度表处理得到正确得料号，同时去重、合并历史数据
- ADS： 根据业务部门需求，生成业务数据表

3. 作业实现

3.0.1. 料号匹配表

> 该数据暂未上线，只有一个测试版本，为暂定作业

```
import datetime
import calendar
import time

import pandas as pd
import pyarrow.parquet as pq
from azure.storage.blob.blockblobservice import BlockBlobService
```

```
from azure.storage.blob import ContainerClient
import json
import os
import sys
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, HiveContext


def get_blob_clib(blobContainName):
    """获取blob客户端以及所需目录下的blob名称"""
    connection_string =
"DefaultEndpointsProtocol=https;AccountName=proddataplatcn3blob01;AccountKey=ScGueSagWl9
s5XDCJeE6xOD8CupGFi5Jp0m9ZVi1Ri812p2GtXD5AXQ/zsVFIcUrNRE2zrIZlWLCjORJZyZHbQ==;EndpointSu
ffix=core.chinacloudapi.cn"

    container = ContainerClient.from_connection_string(
        conn_str=connection_string,
        container_name=blobContainName)
    return container


class Azure_blob():
    """数据模型化"""

    def __init__(self, data_list, container):
        self.data_list = data_list
        self.container = container
        # self.blobContainName =blobContainName

    def get_spark_connection(self):
        spark = SparkSession.builder.config(conf=SparkConf().setAppName("pyspark—to-
hive").set(
            "spark.hadoop.mapreduce.fileoutputcommitter.marksuccessfuljobs",
"false").set("spark.driver.memory", "4g").set("spark.executor.memory",
"4g")).enableHiveSupport().getOrCreate()
        sc = spark.sparkContext
        hc = HiveContext(sc)
        hc.setConf("hive.metastore.warehouse.dir", "/boschfs/warehouse/azure_blob/")
        return spark

    def get_blob_service(self):
        """下载连接"""
        return BlockBlobService(account_name='proddataplatcn3blob01',

 account_key='ScGueSagWl9s5XDCJeE6xOD8CupGFi5Jp0m9ZVi1Ri812p2GtXD5AXQ/zsVFIcUrNRE2zrIZlW
LCjORJZyZHbQ==',
                                endpoint_suffix='core.chinacloudapi.cn')

    def get_blobs(self, num):
        """获取指定容器一级目录下的csv—blobs"""
```

```
        blobs = list(self.container.list_blobs(name_starts_with=(self.data_list[num] +
'/')))
        blobs_list = []
        for i in blobs:
            i = i.name
            if i[-3:] == 'csv':
                blobs_list.append(i)

        return blobs_list

    def get_kehu_code(self, blobs):
        """获取经销商代号"""
        kehu = {a.split('/')[1] for a in blobs}
        if kehu == {}:
            print("==========无经销商==========")
            return 1
        else:
            return kehu

    def get_new_date(self, num, kehu_code):
        """获取指定经销商最新日期和当前的一级目录名"""
        blobs = list(self.container.list_blobs(name_starts_with=(self.data_list[num] +
'/' + kehu_code + '/')))

        da = {a.name.split('/')[2] for a in blobs}
        da2 = []
        print(da)
        for i in da:
            try:
                b = int(i)
                da2.append(b)
            except Exception as e:
                b = 0
                da2.append(b)
        date = max(da2)
        if date == 0 or da == {}:
            print("==========经销商数据未上传==========")
            return 2
        else:
            return date

    def get_DF(self, blob_name,blobContainName):
        """读取blob数据"""

        # 获取blob

        blobs_json = []
        df_list = []

        for b in blob_name:
            # 判断是否空
```

```
        if b == '':
            print('=========经销商未上传数据=============')
            return '3'

        last_index = b.rfind('/')
        # 父级目录信息
        uplt = blobContainName + b[:last_index]
        print(uplt)
        # blob名字
        blob = b[last_index + 1:]
        kehu_code = b.split('/')[1]

        # print(blob)

        # 创建本地文件路径
        blobDirName = os.path.dirname(blob)
        newBlobDirName = os.path.join(uplt, blobDirName)
        if not os.path.exists(newBlobDirName):
            os.makedirs(newBlobDirName)
        localFileName = os.path.join(uplt, blob)

        # 下载
        model.get_blob_service().get_blob_to_path(uplt, blob, localFileName)
        downloadPath = sys.path[0] + "/" + localFileName
        # print(downloadPath)

        # 后缀名
        last_index = blob.rfind('.')
        # blob名字
        blob_end = blob[last_index + 1:]
        # print(blob_end)
        if blob_end == 'json':
            try:
                table = pd.read_json(downloadPath)
                table.rename(columns=str.lower, inplace=True)
                df_list.append(table)
            except Exception as e:
                print(f"=========文件损坏: {downloadPath}==========")
                continue

        # if blob_end == 'parquet':
        elif blob_end == 'parquet':
            try:
                table = pd.read_parquet(downloadPath)
                table.rename(columns=str.lower, inplace=True)
                df_list.append(table)
            except Exception as e:
                print(f"=========文件损坏: {downloadPath}==========")
                continue
        elif blob_end == 'csv':
            try:
```

```
                    table = pd.read_csv(downloadPath)
                    table.rename(columns = str.lower, inplace = True)
                    df_list.append(table)

                except Exception as e:
                    print(f"=========文件损坏: {downloadPath}==========")
                    continue

            else:
                print('======经销商上传文件格式错误=======')
                return '4'


        merge_df = pd.concat(df_list)


        df_merge = merge_df.fillna('')
        return df_merge


    def data_need(self,kehu_code):
        """获取指定经销商的本月blobs和上月月初、月末blobs"""
        blobs = list(container.list_blobs(name_starts_with=('Dealer_stock/' + kehu_code
+ '/')))



        blobs_month = []

        for i in blobs:
            da = i.name
            # 过滤模板
            try:
                b = int(da.split('/')[2])
            except Exception as e:
                continue

            blobs_month.append(da)


        return blobs_month



    def insert_target_table(self, df, database, data, tmp_table, schema):


        # 删表
        sql1 = """
        DROP TABLE IF EXISTS {0}.ods_dealer_product_code_mapping_{1}_df
        """.format(database, data)
```

```
        sql = """
        ALTER TABLE {0}.ods_dealer_product_code_mapping_{1}_df DROP IF EXISTS PARTITION
(ds=${bdp.system.bizdate})
        """.format(database, data)


        # 建表
        sql2= """
         create table if not exists {0}.ods_dealer_product_code_mapping_{1}_df (
            {2}
        )
        partitioned by (ds string) stored as parquet
        location
"boschfs://boschfs/warehouse/{0}.ods_dealer_product_code_mapping_{1}_df"
        """.format(database, data,schema)

        sparkConn.sql(sql1)

        sparkConn.sql(sql2)


        # 测试上传列与hive表已有列是否匹配
        test = """
        select * from {0}.ods_dealer_product_code_mapping_{1}_df limit 1
        """.format(database, data)

        t=sparkConn.sql(test)

        for col in list(df.columns):
            if col not in list(t.columns) and col != 'ds' and col != 'id':
                print('1')
                sql3="""
                ALTER TABLE {0}ods_dealer_product_code_mapping_{1}_df add columns ({2}
STRING comment '')
                """.format(database, data,col)
                sparkConn.sql(sql3)







        for col in list(t.columns):
            if col not in list(df.columns) and col != 'ds' and col != 'id':
                df[col] = 'NULL'
        print(df.columns)
        print(t.columns)

        t=sparkConn.sql(test)

        schema1 = ','.join(t.columns).replace(',ds', '')
```

```
        df = sparkConn.createDataFrame(df)
        df.createOrReplaceTempView(tmp_table)


        sql4 = """
        INSERT overwrite TABLE {0}.ods_dealer_product_code_mapping_{1}_df PARTITION
(ds=${bdp.system.bizdate})
        select {2} from {3}
        """.format(database, data,schema1,tmp_table)

        sql5 = """
        alter table {0}.ods_dealer_product_code_mapping_{1}_df drop partition(ds <
${yyyyMMdd, -7d});
        """.format(database, data)


        sparkConn.sql(sql4)
        # sparkConn.sql(sql5)
        sparkConn.catalog.dropTempView("tmp_table")

    def finish_etl(self,blobs_list,kehu):
        """根据给的一级目录名，读取完成对应的json&parquet数据文件导入"""
        # 临时表
        tmp_table = kehu + '_df_tmp'
        df_list = []

        # 读取合并处理过的json和parquet下载地址列表
        merge_df = model.get_DF(blobs_list,blobContainName)
        if type(merge_df) != str:
            df_list.append(merge_df)
        # 整合所有经销商数据
        fin_merge = pd.concat(df_list)
        print(fin_merge)


        col = list(fin_merge.columns)
        print(col)
        for col_name in col:
            fin_merge[col_name] = fin_merge[col_name].astype(str)

        schema = ' string,'.join(col) + ' string'

        model.insert_target_table(fin_merge, database, kehu, tmp_table, schema)
        time.sleep(2)

if __name__ == '__main__':
    data_list1 = ['Dealer_forecast', 'Dealer_stock', 'test11']
```

```
    blobContainName = 'bosch-data-warehouse/'
    database = 'boschpro'

    # 读取列表参数
    b2 = 2

    # todo 0: 连接blob
    container = get_blob_clib(blobContainName)
    model = Azure_blob(data_list1, container)

    # 创建sparksession
    sparkConn = model.get_spark_connection()
    sparkConn.sparkContext.setLogLevel("Error")

    #   todo 2:指定经销商数据blob名读取
    kehu_business = model.get_blobs(b2)

    # 获取当中csv文件
    kehu = '118002298'
    model.finish_etl(kehu_business,kehu)
```

3.0.2. 料号整合表

```
"""
--*********************************************************************--
--所属主题: 库存域
--功能描述: 经销商库存数据料号mapping merge
--创建者: 王兆翔
--创建日期:2023-05-10
--*********************************************************************--
"""
import pandas as pd
import os
import sys
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, HiveContext


def get_spark_connection():
    """创建spark对象"""
    spark = SparkSession.builder.config(conf=SparkConf().setAppName("pyspark-to-
hive").set(
```

```
        "spark.hadoop.mapreduce.fileoutputcommitter.marksuccessfuljobs",
"false").set("spark.driver.memory", "4g").set("spark.executor.memory",
"4g")).enableHiveSupport().getOrCreate()
    sc = spark.sparkContext
    hc = HiveContext(sc)
    hc.setConf("hive.metastore.warehouse.dir", "/boschfs/warehouse/azure_blob/")
    return spark

def insert_data(df, database, data, tmp_table, schema):
    """数据与hive进行交互"""



     # 删表
    sql1 = """
    DROP TABLE IF EXISTS {0}.{1}
    """.format(database, data)



    # 建表
    sql2= """
     create table if not exists {0}.{1} (
        {2}
    )
    partitioned by (ds string) stored as parquet
    location "boschfs://boschfs/warehouse/{0}.{1}"
    """.format(database, data,schema)

    sparkConn.sql(sql1)

    sparkConn.sql(sql2)


    # 测试上传列与hive历史表已有列是否匹配
    test = """
    select * from {0}.{1} limit 1
    """.format(database, data)

    t=sparkConn.sql(test)

    for col in list(df.columns):
        if col not in list(t.columns) and col != 'ds':
            sql3="""
            ALTER TABLE {0}.{1} add columns ({2} STRING comment '')
            """.format(database, data,col)
            sparkConn.sql(sql3)
```

```
    for col in list(t.columns):
        if col not in list(df.columns) and col != 'ds':
            df[col] = 'NULL'
    print(df.columns)
    print(t.columns)

    schema1 = ','.join(t.columns).replace(',ds', '').replace(',id', '')
    print(schema1)


    df = sparkConn.createDataFrame(df)
    df.createOrReplaceTempView(tmp_table)


    sql4 = """
    INSERT overwrite TABLE {0}.{1} PARTITION (ds=${bdp.system.bizdate})
    select {2} from {3}
    """.format(database, data,schema1,tmp_table)

    sql5 = """
    alter table {0}.{1} drop partition(ds < ${yyyyMMdd, -7d});
    """.format(database, data)


    sparkConn.sql(sql4)
    # sparkConn.sql(sql5)
    sparkConn.catalog.dropTempView("tmp_table")


if __name__ == '__main__':

    database = 'boschpro'
    table = f'dwd_del_dealer_product_code_map_df'
    tmp_table = table + '_tmp_${bdp.system.bizdate}'



    sparkConn = get_spark_connection()
    sparkConn.sparkContext.setLogLevel("Error")


    sql2 = """
    show tables in {0} like 'ods_dealer_product_code_mapping_*'
    """.format(database)
    df=sparkConn.sql(sql2)
    df.show()

    # 获取ods对应的数据表名
    table_list = []
```

```
for i in df.collect():
    print(i)
    table_name = i.tableName
    table_list.append(table_name)

# 获取合并的df列表

df_list = []
print(table_list)


# 取出数据合并
for i in table_list:
    kehu = i.split('_')[5]

    sql = """
    select * from {0}.{1} where ds = '${bdp.system.bizdate}'
    """.format(database,i)

    df = sparkConn.sql(sql)
    df = df.toPandas()
    df['customercode'] = kehu
    df.rename(columns=str.lower, inplace=True)
    df_list.append(df)


df_merge = pd.concat(df_list)
print(df_merge)

col = list(df_merge.columns)
print(col)

for col_name in col:
    df_merge[col_name] = df_merge[col_name].astype(str)

col.remove('ds')
schema = ' string,'.join(col) + ' string'

insert_data(df_merge, database, table, tmp_table, schema)
```

3.0.3. 经销商数据整合

> - ODS auto采集各个经销商数据形成数据表，merge为每日整合表，经过料号匹配成为 “经销商数据增量整合表”，
> - 同时插入历史数据表，对历史数据表进行merge和去重，得到“经销商历史数据全量整合表”
> - 经销商历史数据全量整合表生命周期暂为2，保证其数据备份

3.0.3.1. 经销商auto采集作业

- 经销商预测数据和库存数据每日抽取最近一天

```
"""
--*********************************************************************--
--所属主题：库存域
--功能描述：经销商库存&预测数据，Auto ETL
--创建者：王兆翔
--创建日期:2023-05-12
--*********************************************************************--
"""
import pandas as pd
import pyarrow.parquet as pq
from azure.storage.blob.blockblobservice import BlockBlobService
from azure.storage.blob import ContainerClient
import json
import os
import sys
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, HiveContext


def get_blob_clib(blobContainName):
    """获取blob客户端以及所需目录下的blob名称"""
    connection_string =
"DefaultEndpointsProtocol=https;AccountName=proddataplatcn3blob01;AccountKey=ScGueSagWl9
s5XDCJeE6xOD8CupGFi5Jp0m9ZVi1Ri812p2GtXD5AXQ/zsVFIcUrNRE2zrIZlWLCjORJZyZHbQ==;EndpointSu
ffix=core.chinacloudapi.cn"

    container = ContainerClient.from_connection_string(
        conn_str=connection_string,
        container_name=blobContainName)
    return container


class Azure_blob():
    """数据模型化"""

    def __init__(self, data_list, container):
        self.data_list = data_list
        self.container = container
        # self.blobContainName =blobContainName

    def get_spark_connection(self):
        spark = SparkSession.builder.config(conf=SparkConf().setAppName("pyspark-to-
hive").set(
```

```
            "spark.hadoop.mapreduce.fileoutputcommitter.marksuccessfuljobs",
"false")).enableHiveSupport().getOrCreate()
        sc = spark.sparkContext
        hc = HiveContext(sc)
        hc.setConf("hive.metastore.warehouse.dir", "/boschfs/warehouse/azure_blob/")
        return spark

    def get_blob_service(self):
        """下载连接"""
        return BlockBlobService(account_name='proddataplatcn3blob01',

 account_key='ScGueSagWl9s5XDCJeE6xOD8CupGFi5Jp0m9ZVi1Ri812p2GtXD5AXQ/zsVFIcUrNRE2zrIZlW
LCjORJZyZHbQ==',
                                endpoint_suffix='core.chinacloudapi.cn')

    def get_blobs(self, num):
        """获取指定容器一级目录下的blobs名称"""
        blobs = list(self.container.list_blobs(name_starts_with=(self.data_list[num] +
'/')))
        blobs_list = []
        for i in blobs:
            i = i.name
            blobs_list.append(i)
        return blobs_list

    def get_kehu_code(self, blobs):
        """获取经销商代号"""
        kehu = {a.split('/')[1] for a in blobs}
        if kehu == {}:
            print("==========无经销商============")
            return 1
        else:
            return kehu

    def get_new_date(self, num, kehu_code):
        """获取指定经销商最新日期和当前的一级目录名"""
        blobs = list(self.container.list_blobs(name_starts_with=(self.data_list[num] +
'/' + kehu_code + '/')))

        da = {a.name.split('/')[2] for a in blobs}
        da2 = []
        print(da)
        for i in da:
            try:
                b = int(i)
                da2.append(b)
            except Exception as e:
                b = 0
                da2.append(b)
        date = max(da2)
        if date == 0 or da == {}:
```

```
            print("==========经销商数据未上传============")
            return 2
        else:
            return date

    def get_file(self, num, kehu, date, blobContainName):
        """读取合并指定日期下所有json数据 或者读取 单个parquet文件"""
        blob_name = list(self.container.list_blobs(
            name_starts_with=(self.data_list[num] + '/' + kehu + '/' + str(date) +
'/')))
        print(self.data_list[num] + '/' + kehu + '/' + str(date) + '/')

        # 获取blob
        blobs_json = []
        blobs_parquet = []
        for b in blob_name:
            b = b.name
            # 判断是否空
            if b == '':
                print('=========经销商未上传数据=============')
                return '3','3'

            last_index = b.rfind('/')
            # 父级目录信息
            uplt = blobContainName + b[:last_index]
            print(uplt)
            # blob名字
            blob = b[last_index + 1:]
            print(blob)

            # 创建本地文件路径
            blobDirName = os.path.dirname(blob)
            newBlobDirName = os.path.join(uplt, blobDirName)
            if not os.path.exists(newBlobDirName):
                os.makedirs(newBlobDirName)
            localFileName = os.path.join(uplt, blob)

            # 下载
            model.get_blob_service().get_blob_to_path(uplt, blob, localFileName)
            downloadPath = sys.path[0] + "/" + localFileName

            # 后缀名
            last_index = blob.rfind('.')
            # blob名字
            blob_end = blob[last_index + 1:]
            # print(blob_end)
            if blob_end == 'json':
                # 打开JSON文件并解析数据
                with open(downloadPath, encoding='utf8') as f:
                    json_load = json.load(f)
                    for i in json_load:
```

```
                    blobs_json.append(i)
                # 将解析后的数据合并到merged_data中关闭
                f.close()

            elif blob_end == 'parquet':

                blobs_parquet.append(downloadPath)

            else:
                print('======经销商上传文件格式错误=======')
                return '4','4'
        # 列表转json
        json_str = json.dumps(blobs_json)

        # 单个地址返回
        if blobs_parquet != []:
            for i in blobs_parquet:
                return '', i
        else:
            return json_str,''


    def insert_target_table(self, df, database, data, kehu, tmp_table, schema):
        """建表与插入"""

        # 删表

        sql1 = """
        DROP TABLE IF EXISTS {0}.ods_azure_blob_auto_{1}_{2}_df
        """.format(database, data, kehu)


        # 建表
        sql2= """
         create table if not exists {0}.ods_azure_blob_auto_{1}_{2}_df (
            {3}
        )
        partitioned by (ds string) stored as parquet
        location "boschfs://boschfs/warehouse/{0}.ods_azure_blob_auto_{1}_{2}_df"
        """.format(database, data, kehu, schema)



        # sparkConn.sql(sql1)
        sparkConn.sql(sql2)


        # 测试上传列与hive表已有列是否匹配
        test = """
        select * from {0}.ods_azure_blob_auto_{1}_{2}_df limit 1
        """.format(database, data, kehu, schema)
```

```
        t=sparkConn.sql(test)

        for col in list(df.columns):
            if col not in list(t.columns) and col != 'ds':
                print('1')
                sql3="""
                ALTER TABLE {0}.ods_azure_blob_auto_{1}_{2}_df add columns ({3} STRING
comment '')
                """.format(database, data, kehu,col)
                sparkConn.sql(sql3)




        for col in list(t.columns):
            if col not in list(df.columns) and col != 'ds':
                df[col] = 'NULL'
        print(df.columns)
        print(t.columns)

        t=sparkConn.sql(test)

        schema1 = ','.join(t.columns).replace(',ds', '')

        # 插入数据
        df = sparkConn.createDataFrame(df)
        df.show()
        df.createOrReplaceTempView(tmp_table)



        sql4 = """
        INSERT OVERWRITE TABLE {0}.ods_azure_blob_auto_{1}_{2}_df PARTITION
(ds=${bdp.system.bizdate})
        select {3} from {4}
        """.format(database, data, kehu,schema1,tmp_table)


        sparkConn.sql(sql4)



    def finish_etl(self, num, kehu_logo):
        """根据给的一级目录名，读取完成对应的json&parquet数据文件导入"""
        for kehu_code in kehu_logo:
            date_kehu = model.get_new_date(num, kehu_code)
            # 临时表
            tmp_table = data_list1[num] + '_' + kehu_code + '_df_tmp'
```

```
            # 读取合并处理过的json和parquet下载地址列表
            json_str, parquet_str = model.get_file(num, kehu_code, date_kehu,
blobContainName)
            print(parquet_str)

            # 如果有危机
            if json_str == '3' or json_str == '4':
                continue

            elif json_str != '':
                # 将JSON字符串转换为DataFrame
                df_json = pd.read_json(json_str)


                # # df.columns = col
                df_json = df_json.fillna('')

                keys = list(df_json.keys())
                # print(keys)
                schema = ' string,'.join(keys)+' string'



                model.insert_target_table(df_json, database, data_list1[num], kehu_code,
tmp_table, schema)
                # df.stop()

            elif parquet_str != '':
                # 读取Parquet文件的架构
                schema = pq.read_schema(parquet_str)
                # print(schema)

                # 获取列名
                col = list(schema.names)
                schema = ' string,'.join(col)+' string'

                df = pd.read_parquet(parquet_str)


                df.columns = col
                for col_name in col:
                    df[col_name] = df[col_name].astype(str)
                # print(df)



                model.insert_target_table(df, database, data_list1[num], kehu_code,
tmp_table, schema)
            else:
                print('========未读取到json和parquet文件，请检查=========')
```

```
if __name__ == '__main__':
    data_list1 = ['Dealer_forecast', 'Dealer_stock']
    blobContainName = 'bosch-data-warehouse/'
    database = 'boschpro'

    # 读取列表参数
    p1 = 0
    b2 = 1

    # todo 0: 连接blob
    container = get_blob_clib(blobContainName)
    model = Azure_blob(data_list1, container)

    # 创建sparksession
    sparkConn = model.get_spark_connection()
    sparkConn.sparkContext.setLogLevel("Error")

    #   todo 2:指定经销商数据blob名读取
    kehu_predict = model.get_blobs(p1)
    kehu_business = model.get_blobs(b2)



    # todo 3: 获取经销商代号列表
    kehu_predict_code = model.get_kehu_code(kehu_predict)
    kehu_business_code = model.get_kehu_code(kehu_business)
    print(kehu_predict_code)

    # todo 4:进行etl
    predict = model.finish_etl(p1,kehu_predict_code)
    business = model.finish_etl(b2,kehu_business_code)
```

3.0.3.2. 经销商库存整合作业

- 将所有经销商库存数据进行整合

```
"""
--*********************************************************************--
--所属主题：库存域
--功能描述：进销商库存数据，经销商代码
--创建者：王兆翔
--创建日期:2023-05-10
--*********************************************************************--
"""
import pandas as pd
```

```
import os
import sys
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession, HiveContext


def get_spark_connection():
    """创建spark对象"""
    spark = SparkSession.builder.config(conf=SparkConf().setAppName("pyspark-to-
hive").set(
        "spark.hadoop.mapreduce.fileoutputcommitter.marksuccessfuljobs",
"false").set("spark.driver.memory", "4g").set("spark.executor.memory",
"4g")).enableHiveSupport().getOrCreate()
    sc = spark.sparkContext
    hc = HiveContext(sc)
    hc.setConf("hive.metastore.warehouse.dir", "/boschfs/warehouse/azure_blob/")
    return spark

def insert_data(df, database, data, tmp_table, schema):
    """数据与hive进行交互"""

    df = sparkConn.createDataFrame(df)
    df.createOrReplaceTempView(tmp_table)

     # 删表
    sql1 = """
    DROP TABLE IF EXISTS {0}.ods_azure_blob_auto_{1}_di
    """.format(database, data)


    # 建表
    sql2= """
     create table if not exists {0}.ods_azure_blob_auto_{1}_di (
        {2}
    )
    partitioned by (ds string) stored as parquet
    location "boschfs://boschfs/warehouse/{0}.ods_azure_blob_auto_{1}_di"
    """.format(database, data,schema)


    sql4 = """
    INSERT overwrite TABLE {0}.ods_azure_blob_auto_{1}_di PARTITION
(ds=${bdp.system.bizdate})
    select {2} from {3}
    """.format(database, data,schema,tmp_table)



    sparkConn.sql(sql1)
    sparkConn.sql(sql2)
    sparkConn.sql(sql4)
```

```
    sparkConn.catalog.dropTempView("tmp_table")


if __name__ == '__main__':

    database = 'boschpro'
    data = 'dealer_stock'
    table = f'ods_azure_blob_auto_{data}_di'
    tmp_table = table + '_tmp_${bdp.system.bizdate}'



    sparkConn = get_spark_connection()
    sparkConn.sparkContext.setLogLevel("Error")


    sql2 = """
    show tables in {0} like 'ods_*auto_dealer_stock_*_df'
    """.format(database)
    df=sparkConn.sql(sql2)
    df.show()

    # 获取ods对应的数据表名
    table_list = []
    print(df)
    for table in df.collect():
        print(table)
        table_name = table.tableName
        table_list.append(table_name)

    # 获取合并的df列表

    df_list = []


    # 取出数据合并
    for table in table_list:
        kehu = table.split('_')[6]

        sql = """
        select * from {0}.{1} where ds = '${bdp.system.bizdate}'
        """.format(database,table)

        df = sparkConn.sql(sql)
        df = df.toPandas()
        df['kehu'] = kehu
        df.rename(columns=str.lower, inplace=True)
        df_list.append(df)
```

```
    df_merge = pd.concat(df_list)

    col = list(df_merge.columns)
    print(col)

    for col_name in col:
      df_merge[col_name] = df_merge[col_name].astype(str)

    col.remove('ds')
    schema = ' string,'.join(col) + ' string'

    insert_data(df_merge, database, data, tmp_table, schema)
```

3.0.3.3. 经销商库存增量处理作业

- 对整合的数据进行料号匹配，并插入增量表和全量表之中——料号处理逻辑见需求说明

```
--*************************************************************--
--所属主题：库存域
--功能描述：进销商库存每日增量数据插入
--创建者：王兆翔
--创建日期:2023-05-24
--*************************************************************--
set hive.tez.container.size=4096;

with t1 as (
    -- alternative number 通用料号匹配表
    select
        material,
        comp_alt_no
    from(
            select
                row_number() OVER(PARTITION by comp_alt_no) group_distinct,
                material, --材料号
                comp_alt_no --比较选择料号
            from
                ods_azure_blob_v_ymtk00101_df y
            left semi join
                (select max(ds) ds from ods_azure_blob_v_ymtk00101_df) o
            on y.ds = o.ds
        ) t
    where
        group_distinct = 1
)
, t2 as(
    -- cross reference 原厂编号匹配表
    select
        distinct matnr,
```

```
        khnr_verd
    from
        (
            select
                row_number() over(partition by khnr_verd) as group_distinct,
                matnr, --料号
                khnr_verd   -- 原厂处理后
            from
                ods_azure_blob_ymtk10001_cross_ref_info_df y
            left semi join
                (select max(ds) ds from ods_azure_blob_ymtk10001_cross_ref_info_df) o
            on y.ds = o.ds
        ) t
    where
        group_distinct = 1 and matnr is not null
)
, t3 as (
    -- 全球物料匹配表
    select
        material_10_digits
    from(
            select
                row_number() over(partition by material_10_digits) as group_distinct,
                material_10_digits         -- 10位料号
            from
                ods_azure_blob_v_aamm_gen_material_df y -- 开思物料匹配表
            left semi join
                  (select max(ds) ds from ods_azure_blob_v_aamm_gen_material_df) o
            on y.ds = o.ds
        ) t
    where
        group_distinct = 1
)
, t4 as (
    select
        distinct customercode,
        product_code,
        boschpartno
    from dwd_del_dealer_product_code_map_df
)
from (select * from ods_azure_blob_auto_dealer_stock_di where
ds='${bdp.system.bizdate}') a
left join t1 on a.boschpartno = t1.comp_alt_no
left join t2 on a.boschpartno = t2.khnr_verd
left join t3 on a.boschpartno = t3.material_10_digits
left join t4 on a.kehu = t4.customercode and a.boschpartno = t4.product_code

insert overwrite table dwd_dealer_stock_history_df partition(ds =
'${bdp.system.bizdate}')
select
    kehu customercode   -- 客户代码
```

```
    ,warehousename      -- 仓库名称
    ,warehouseno        -- 仓库编号
    ,a.boschpartno      -- 博世10位料号
    ,boschpartno13      -- 博世13位料号
    ,productname        -- 产品描述
    ,productcategory    -- 产品品类
    ,stockqty           -- 库存数量
    ,unit               -- 计量单位
    ,substring(regexp_replace(loaddate,'[^0-9]',''),1,8) loaddate           -- 库存导出日
期
    ,regexp_replace(loadtime,'[^0-9:]','') loadtime                         -- 库存导出时
间
    ,case
        when t4.product_code is not null then t4.boschpartno
        when regexp_replace(substring(a.boschpartno, 0, 10),'[^a-zA-Z0-9]','') =
t3.material_10_digits then t3.material_10_digits
        when regexp_replace(substring(a.boschpartno, 0, 10),'[^a-zA-Z0-9]','') =
t1.comp_alt_no then t1.material
        when regexp_replace(substring(a.boschpartno, 0, 10),'[^a-zA-Z0-9]','') =
t2.khnr_verd then t2.matnr
        else 0
    end as boschpartno  -- 匹配过的博世10位料号

insert overwrite table dwd_latest_dealer_stock_di partition(ds =
'${bdp.system.bizdate}')
select
    kehu customercode   -- 客户代码
    ,warehousename      -- 仓库名称
    ,warehouseno        -- 仓库编号
    ,a.boschpartno      -- 博世10位料号
    ,boschpartno13      -- 博世13位料号
    ,productname        -- 产品描述
    ,productcategory    -- 产品品类
    ,stockqty           -- 库存数量
    ,unit               -- 计量单位
    ,substring(regexp_replace(loaddate,'[^0-9]',''),1,8) loaddate           -- 库存导出日
期
    ,regexp_replace(loadtime,'[^0-9:]','') loadtime                         -- 库存导出时
间
    ,case
        when t4.product_code is not null then t4.boschpartno
        when regexp_replace(substring(a.boschpartno, 0, 10),'[^a-zA-Z0-9]','') =
t3.material_10_digits then t3.material_10_digits
        when regexp_replace(substring(a.boschpartno, 0, 10),'[^a-zA-Z0-9]','') =
t1.comp_alt_no then t1.material
        when regexp_replace(substring(a.boschpartno, 0, 10),'[^a-zA-Z0-9]','') =
t2.khnr_verd then t2.matnr
        else 0
    end as boschpartno  -- 匹配过的博世10位料号
```

```
-- select count(*) from ods_azure_blob_v_aamm_gen_material_df
```

3.0.3.4. 经销商全量数据去重作业

```
--*************************************************************************--
--所属主题：库存域
--功能描述：进销商库存本月数据及其他月月初月末数据
--创建者：王兆翔
--创建日期:2023-06-07
--*************************************************************************--
set hive.tez.container.size = 4096;

insert overwrite table dwd_dealer_stock_history_df partition (ds =
'${bdp.system.bizdate}')
select
    customercode,
    warehousename,
    warehouseno,
    boschpartno,
    boschpartno13,
    productname,
    productcategory,
    stockqty,
    unit,
    loaddate,
    loadtime,
    boschpartno_verified
from (
select
    *,
    row_number() over(partition by
    customercode,
    warehousename,
    warehouseno,
    boschpartno,
    boschpartno13,
    productname,
    productcategory,
    stockqty,
    unit,
    loaddate,
    loadtime,
    boschpartno_verified) rn
from
    dwd_dealer_stock_history_df) a
where rn  = 1
```

3.0.4. 经销商库存月揽表

- 基于已经处理的库存历史整合表进行数据应用和输出

```
--*************************************************************--
--所属主题：库存域
--功能描述：进销商库存本月数据及其他月月初月末数据
--创建者：王兆翔
--创建日期:2023-06-07
--*************************************************************--
set hive.tez.container.size = 4096;

with t1 as (
    select
        customercode,
        warehousename,
        warehouseno,
        boschpartno,
        boschpartno13,
        productname,
        productcategory,
        stockqty,
        unit,
        concat_ws('-'
,substring(loaddate,1,4),substring(loaddate,5,2),substring(loaddate,7,2)) loaddate,
        loadtime,
        boschpartno_verified
    from
        dwd_dealer_stock_history_df
    where ds =  ${bdp.system.bizdate}
) insert overwrite table ads_stock_visualization_dealer_stock_history_df partition(ds =
${bdp.system.bizdate})
select
    customercode,
    warehousename,
    warehouseno,
    boschpartno,
    boschpartno13,
    productname,
    productcategory,
    stockqty,
    unit,
    date_format(loaddate,'yyyyMMdd') loaddate,
    loadtime,
    boschpartno_verified
from t1
```

```
where (month(loaddate) != month(current_date()) and (loaddate = last_day(loaddate) or
day(loaddate) = 1))
   or month(loaddate) = month(current_date())
```

4. 结果输出

- ods_dealer_product_code_mapping_118002298_df
- dwd_del_dealer_product_code_map_df
- dwd_latest_dealer_stock_di
- dwd_dealer_stock_history_df
- ads_stock_visualization_dealer_stock_history_df