MSPA PREDICT 420

Graded Exercise 3: Creating Assets

Introduction

This document presents the results of the third graded exercise for the Masters of Science in Predictive Analytics course: PREDICT 420. This assessment required the student to create tw o data "assets" for use by XYZ company, create a relational database for these assets, and create a "flat" file w ith selected customers and variables.

Assessment

0. Accessing Postgres Server on the SSCC

Login to dornick.it

In [1]:
```
#\ssh -YC xxx@dornick.it.northwestern.edu
```

Login to Postgres server and connect to XYZ

In [2]:
```
#psql -h 129.105.208.226 -U xxx -d postgres
#\c xyz
```

Create temp view of item, mail and customer tables and copy to csv

In [3]:
```
#CREATE TEMP VIEW dgb2583item as SELECT * FROM pilot.item;
#\copy (SELECT * FROM dgb2583item) TO 'item.csv' WITH DELIMITER ',' NULL AS '\null' CSV HEADER

#CREATE TEMP VIEW dgb2583mail as SELECT * FROM pilot.mail;
#\copy (SELECT * FROM dgb2583mail) TO 'mail.csv' WITH DELIMITER ',' NULL AS '\null' CSV HEADER

#CREATE TEMP VIEW dgb2583customer as SELECT * FROM pilot.customer;
#\copy (SELECT * FROM dgb2583customer) TO 'customer.csv' WITH DELIMITER ',' NULL AS '\null' CSV
HEADER
```

1. Loading the Data

Load datasets into pandas dataframes.

In [4]:
```
import pandas as pd

df_customer = pd.read_csv("data/customer.csv") # Blanks with string if desired, na_values = "N.A
."
df_item = pd.read_csv("data/item.csv")
df_mail = pd.read_csv("data/mail.csv")
```

2. Remove Duplicate Records

Find duplicate records w ithin the 'customer' dataframe.

In [5]:
```
df_customer["Duplicate"] = df_customer.duplicated(["acctno"])
print("Duplicate customer records:", len(df_customer[df_customer.Duplicate == True]))
```

```
Duplicate customer records: 0
```

3. Check Customer Record Consistency in Item/Mail Dataframes

Find 'acctno' records w ithin the 'item' dataframe w hich do not appear as 'acctno' records w ithin the 'customer' dataframe.

In [6]:
```
df_item["Matched"] = df_item["acctno"].isin(df_customer["acctno"])
print("Number of item records with no customer:", len(df_item[df_item.Matched == False]))
```

```
Number of item records with no customer: 0
```

Find 'acctno' records within the 'mail' dataframe which do not appear as 'acctno' records within the 'customer' dataframe.

In [7]:
```
df_mail["Matched"] = df_mail["acctno"].isin(df_customer["acctno"])
print("Number of mail records with no customer:", len(df_mail[df_mail.Matched == False]))
```

```
Number of mail records with no customer: 0
```

4. Create SQLite Database and Write Data

In [8]:
```
import itertools, pandas, pandas.io.sql, sqlite3

conn = sqlite3.connect("data/db_pilot.db") # Create SQLite database/connection

list_df = [df_customer, df_item, df_mail]
list_tbl = ["customer", "item", "mail"]

for d, t in zip(list_df, list_tbl): # Loop through each dataframe
    dropQuery = "DROP TABLE IF EXISTS " + str(t) # Drop table if already exists in database
    conn.execute(dropQuery).fetchone()
    pandas.io.sql.to_sql(d, t, conn)
    selQuery = "SELECT * FROM " + str(t) # Import dataframe into database
    conn.execute(selQuery).fetchone()

conn.close() # Close SQLite database connection
```

5. Count Records within each SQLite Database Table

In [9]:
```
import pandas, sqlite3

conn = sqlite3.connect("data/db_pilot.db") # Create SQLite database/connection
curs = conn.cursor()

tblQuery = "SELECT name FROM sqlite_master WHERE type = 'table' ORDER BY Name" # Select all tabl
e query
curs.execute(tblQuery) # Execute select all table query
tables = map(lambda t: t[0], curs.fetchall()) # Fetch list of tables

totalColumns, totalRows, totalCells = 0, 0, 0
resultcols = ["Table Name", "Column Count", "Row (Record) Count", "Cell Count"]

results_df = pd.DataFrame([])

for table in tables:
    columnsQuery = "PRAGMA table_info(%s)" % table # SQLite table info query
    curs.execute(columnsQuery)
    numberOfColumns = len(curs.fetchall()) # Return column count
    rowsQuery = "SELECT Count() FROM %s" % table # Row count query
    curs.execute(rowsQuery)
    numberOfRows = curs.fetchone()[0] # Return row count
    numberOfCells = numberOfColumns * numberOfRows # Derive cell count
    temp_df = pandas.DataFrame([[table, numberOfColumns, numberOfRows, numberOfCells]], columns
= resultcols)
    results_df = results_df.append(temp_df)

curs.close()
conn.close()

results_df.reset_index().drop("index", 1)
```

Out[9]:

	Table Name	Column Count	Row (Record) Count	Cell Count
0	customer	453	50000	22650000
1	item	10	77121	771210
2	mail	19	30946	587974

6. Create XYZ Direct Mail Marketing Dataframe

Create 'drop table' queries for each table.

In [10]:
```
droptempmailQuery = "DROP TABLE IF EXISTS tempmail"
droptempcustQuery = "DROP TABLE IF EXISTS tempcustomer"
droptempcampQuery = "DROP TABLE IF EXISTS campaign"
```

Query to create temporary mail table which includes sum of all mail fields.

In [11]:
```
tempmailQuery = """
CREATE TABLE tempmail AS
SELECT acctno,
       (mail.mail_1 + mail.mail_2 + mail.mail_3 + mail.mail_4 + mail.mail_5 +
       mail.mail_6 + mail.mail_7 + mail.mail_8 + mail.mail_9 + mail.mail_10 +
       mail.mail_11 + mail.mail_12 + mail.mail_13 + mail.mail_14 + mail.mail_15 +
       mail.mail_16) AS mail_total
FROM mail
"""
```

Query to create temporary customer table w hich includes numeric 'zhomeent' and 'zmobav' fields.

In [12]:
```
tempcustQuery = """
CREATE TABLE tempcustomer AS
SELECT acctno,
       ytd_sales_2009,
       ytd_transactions_2009,
       zhomeent,
       CASE zhomeent WHEN "Y" THEN 1 ELSE 0 END AS zhomeent01,
       zmobav,
       CASE zmobav WHEN "Y" THEN 1 ELSE 0 END AS zmobav01
FROM customer
"""
```

Query to create campaign table w hich includes fields from above tables, joined by account number.

In [13]:
```
campQuery = """
CREATE TABLE campaign AS
SELECT tempmail.acctno AS acctno,
       tempmail.mail_total AS mailCampaigns,
       tempcustomer.ytd_sales_2009 AS YTD09transSum,
       tempcustomer.ytd_transactions_2009 AS YTD09salesSum,
       tempcustomer.zhomeent AS ZHOMEENT,
       tempcustomer.zhomeent01 AS ZHOMEENT01,
       tempcustomer.zmobav AS ZMOBAV,
       tempcustomer.zmobav01 AS ZMOBAV01
FROM tempmail
JOIN tempcustomer ON tempcustomer.acctno = tempmail.acctno
WHERE tempmail.mail_total >= 5
GROUP BY tempmail.acctno
"""
```

Run each query and export created campaign table as csv file.

In [14]:
```
#Note: SQLite does not support SELECT INTO
import pandas, sqlite3

conn = sqlite3.connect("data/db_pilot.db") # Create SQLite database/connection
curs = conn.cursor()

temp = conn.execute(droptempmailQuery) # drop tempmail if exists
temp = conn.execute(droptempcustQuery) # drop tempcustomer if exists
temp = conn.execute(droptempcampQuery) # drop campaign if exists

temp = conn.execute(tempmailQuery) # create tempmail
temp = conn.execute(tempcustQuery) # create tempcustomer

temp = conn.execute(campQuery) # create campaign

exp_df =  pandas.io.sql.read_sql("SELECT * FROM campaign", conn)
if "index" in exp_df:
    exp_df = exp_df.drop("index", 1)

exp_file = "data/campaign.csv"
exp_df.to_csv(exp_file, index = False)

curs.close()
conn.close()
```

7. Confirm Campaign Output

Load dataset into pandas dataframes.

In [15]:
```
import pandas as pd

df_campaign = pd.read_csv("data/campaign.csv")
```

Print last five records of dataframe.

In [16]:
```
df_campaign[-5:]
```

Out[16]:

	acctno	mailCampaigns	YTD09transSum	YTD09salesSum	ZHOMEENT	ZHOMEENT01	ZMOBAV	ZMOBAV01
11387	YYWSAQLL	7	336	1	U	0	U	0
11388	YYYGSWDG	11	801	1	U	0	U	0
11389	YYYHGQYW	10	0	0	U	0	U	0
11390	YYYWSGGA	8	249	1	U	0	U	0
11391	YYYWYPDH	5	0	0	U	0	U	0

Count number of records within dataframe.

In [17]:
```
print("Number of records within campaign dataframe:", len(df_campaign))
```

```
Number of records within campaign dataframe: 11392
```

Confirm recoding for 'ZHOMEENT' and 'ZMOBAV' fields.

In [18]:
```
import pandas as pd

pd.crosstab(df_campaign.ZHOMEENT, df_campaign.ZHOMEENT01, margins = True) # Blanks filled with NaN
```

Out[18]:

ZHOMEENT01	0	1	All
ZHOMEENT			
U	8586	0	8586
Y	0	1909	1909
All	9483	1909	11392

In [19]:
```
import pandas as pd

pd.crosstab(df_campaign.ZMOBAV, df_campaign.ZMOBAV01, margins = True) # Blanks filled with NaN
```

Out[19]:

ZMOBAV01	0	1	All
ZMOBAV			
U	10166	0	10166
Y	0	329	329
All	11063	329	11392

Pickle final dataframe.

In [20]:
```
import pickle

df_campaign.to_pickle("data/campaign.p")
```