**FanDuel Application- David Hannon**

Part 2- SQL Exercise

1. Given the 4 data tables at our disposal, I believe the most important variables for creating a segmented offer (other than primary keys for JOINs) are deposit\_count and account\_suspended from the User table, entry\_date and entry\_fee from the Entry table, and sport and size from the Game table.

Why they are important:

* sport: So we can group users that do not typically play NFL games and may need extra incentive to try vs. users that regularly play NFL games
* size: So that we can group users that prefer head to head games (to incentivize to try larger game sizes as well) and users that prefer larger tournaments (to incentivize them to try head to head games as well and ideally increase their odds of winning)
* deposit\_count: Allows us to group users by those who continually add money to their account or users who have only deposited money once or twice
* account\_suspended: So that we may ignore any suspended accounts
* entry\_date: So that we can group users that are recently active and users who haven’t played in awhile and may need extra incentive to reactivate
* entry\_fee: So that we can group “big spenders”

The groups that I would most like to create would be based on sport and size. For ‘sport’, there would be a group who historically have not tried NFL games much and a group who historically have, so we could give a much greater incentive to the non-NFL group (something like a free-entry match in fantasy or a free bet-match in the sportsbook) and then a lesser ‘welcome back’ incentive to the group who already bet on NFL (like a half price entry fee or parlay insurance bet). This segmentation could be created from setting either NFL game entry count fences or segmented by percentage of total games played that are NFL. Similar incentives could be done if users were grouped based on whether they tend to play head to head games or large pool games to incentivize them to try the other type, assuming they will still play whichever game type they originally preferred as well.

1. To get a count of users who have made at least one deposit and registered in 2013 (Assuming the described USER table has been imported or entered into SQL as ‘DuelUser’):

SELECT COUNT(\*) FROM DuelUser WHERE deposit\_count >=1 AND Year(registration\_date)=2013;

An additional constraint that may be worth adding is a filter on account\_suspended to ensure no suspended accounts are counted in this total (simply add “AND account\_suspended=0”).

1. To add these additional restrictions will require a lengthier code as we now must use joins to combine information from the User, Entry, and Game Tables. The following assumes data is saved in tables labelled DuelUser, DuelEntry, Duel Game, and DuelPayment. Some steps in this logic are also setting us up for the desired table in Question 4.

#Step 1- Join All Info from Game table with the entry table

CREATE TABLE GameEntry AS

SELECT

DuelEntry.\*,

DuelGame.sport,

DuelGame.size

FROM DuelEntry

LEFT JOIN DuelGame on DuelEntry.game\_id=DuelGame.game\_id;

#Step 2- Create a net deposit payment on the payment table to be joined with User table to calculate each users net deposit (in prep for Question 4).

#Adjust amount based on W or D

CREATE TABLE DuelPaymentsAdj AS

SELECT\*,

CASE WHEN payment\_type = 'W' THEN -amount ELSE amount END AS adjusted\_amount FROM DuelPayments;

#Sum NetDeposit per user

CREATE TABLE NetDeposits AS

SELECT user\_id, SUM(adjusted\_amount) AS net\_deposit

FROM DuelPaymentsAdj GROUP BY user\_id;

#Join with User table as User Payments

CREATE TABLE UserPayments AS

SELECT

DuelUser.\*,

NetDeposits.net\_deposit

FROM DuelUser

LEFT JOIN NetDeposits on DuelUser.user\_id = NetDeposits.user\_id;

#Step3- Combine GameEntry and User Payment to compile all pertinent info

CREATE TABLE MasterInfo AS

SELECT

GameEntry.\*,

UserPayments.username,

UserPayments.email,

UserPayments.deposit\_count,

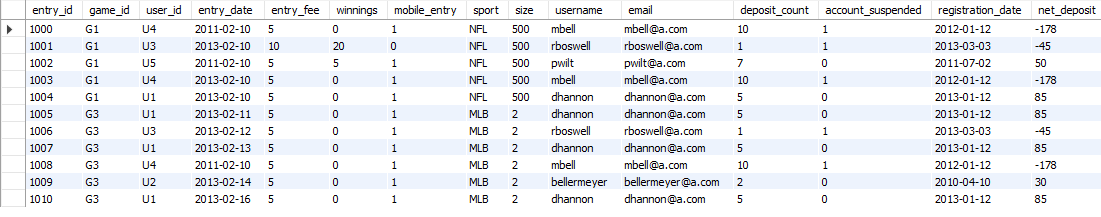
UserPayments.account\_suspended,

UserPayments.registration\_date,

UserPayments.net\_deposit

FROM GameEntry

LEFT JOIN UserPayments on GameEntry.user\_id=UserPayments.user\_id;

#Image of what MasterInfo looks like with my made-up test data:

#Final Step- Filter for distinct users that meet conditions

SELECT COUNT(distinct user\_id) FROM MasterInfo

WHERE deposit\_count >=1

AND Year(registration\_date)=2013

AND sport ='NFL'

AND YEAR(entry\_date)=2013;

#For dummy data, this count yields 2, as both entry\_id 1001 and 1004 satisfy these conditions so the 2 users U3 and U1 are counted.



1. To create the desired list, a few more columns will be joined to the MasterInfo table to make the calculations easier- specifically a breakdown of each users 2013 NFL fees vs. others and each users mobile entry counts. Then, a table of just eligible\_users will be created based on the CRM’s restrictions from question 3. Then the final master info table will be edited to the desired columns listed in question 4. This will then be left joined to the eligible\_users list so that we only see those that met the criteria (users U1 and U3 in the example data).

#Step 1- Add info to MasterInfo Table regarding Fee breakdown of NFL2013 vs all other sports in 2013.

CREATE TABLE MasterInfo2 AS

SELECT\*,

CASE WHEN sport = 'NFL' AND YEAR(entry\_date)=2013 THEN entry\_fee ELSE 0 END AS NFL2013\_Fees,

CASE WHEN sport != 'NFL' AND YEAR(entry\_date)=2013 THEN entry\_fee ELSE 0 END AS Other2013\_Fees

FROM MasterInfo;

#Step 2- Add In Mobile Entry Info per User (for Question 4)

CREATE TABLE MobileEntryInfo AS

SELECT

user\_id,

sum(CASE WHEN mobile\_entry=1 THEN 1 ELSE 0 END) AS UserMobileEntries,

count(\*) AS TotalUserEntries

FROM MasterInfo2

GROUP BY user\_id;

SELECT\*FROM MobileEntryInfo;

CREATE TABLE MasterInfo3 AS

SELECT

MasterInfo2.\*,

MobileEntryInfo.UserMobileEntries,

MobileEntryInfo.TotalUserEntries

FROM MasterInfo2

LEFT JOIN MobileEntryInfo on MasterInfo2.user\_id=MobileEntryInfo.user\_id;

#Step 3- Create Table of just eligible users from question 3 to be left joined the desired info from the CRM Manager.

CREATE TABLE Eligible\_Users AS

SELECT

distinct(user\_id) FROM MasterInfo3

WHERE deposit\_count >=1

AND Year(registration\_date)=2013

AND sport ='NFL'

AND YEAR(entry\_date)=2013;

#Showing this table aligns with the count in Question 3 of the two users, U1 and U3 being eligible

Table

Description automatically generated

##Step 4- Create table of desired info from MasterInfo3 to be joined with Eligible Users

CREATE TABLE CRM AS

SELECT

user\_id,

email,

sum(NFL2013\_Fees) AS 2013NFLFees,

sum(Other2013\_Fees)AS 2013OtherFees,

(UserMobileEntries)/(TotalUserEntries) AS PercentMobile,

MAX(CASE WHEN entry\_fee>0 THEN entry\_date ELSE 'N/A' END) AS LastPaidEntry,

net\_deposit

FROM MasterInfo3

GROUP BY user\_id, email, PercentMobile, net\_deposit;

#Results of CRM table

Table

Description automatically generated

#Step 5- Left Join to Eligible\_Users to filter to only users that satisfy specified criteria

CREATE TABLE CRMFinal AS

SELECT

CRM.\*

FROM Eligible\_Users

LEFT JOIN CRM on Eligible\_Users.user\_id=CRM.user\_id;

#Final Result from dummy data



1. To create a table with years 2011-2013 and their respective percentages of head to head games and head to head fees, a few intermediate tables were created: one of yearly overall totals, one of yearly totals filtered to just head to head games, a table to combine these tables, and finally a table to calculate the Marketing VP’s desired result:

#Step 1- Create table of totals of all games by year for new users (30 days)

CREATE TABLE YearTotals AS

SELECT

YEAR(entry\_date) AS EntryYear,

count(\*) AS YearTotalNewUserEntries,

sum(entry\_fee) AS YearTotalNewUserFees

FROM MasterInfo3

WHERE DATEDIFF(entry\_date, registration\_date)<30

GROUP BY EntryYear;

#Step 2- Create table of yearly Head2Head totals for new users

CREATE TABLE YearH2HTotals AS

SELECT

YEAR(entry\_date) AS EntryYear,

count(\*) AS YearTotalNewUserH2HEntries,

sum(entry\_fee) AS YearTotalNewUserH2HFees

FROM MasterInfo3

WHERE DATEDIFF(entry\_date, registration\_date)<30 AND size=2

GROUP BY EntryYear;

SELECT\*FROM YearH2HTotals;

#Step 3- Combine tables

CREATE TABLE H2HCalc AS

SELECT

YearH2HTotals.\*,

YearTotals.YearTotalNewUserEntries,

YearTotals.YearTotalNewUserFees

FROM YearH2HTotals

LEFT JOIN YearTotals on YearH2HTotals.EntryYear=YearTotals.EntryYear;

#Step 4- Create Final Table of VP's specified output

CREATE TABLE MarketingVPH2H AS

SELECT

EntryYear,

YearTotalNewUserH2HEntries/YearTotalNewUserEntries,

YearTotalNewUserH2HFees/YearTotalNewUserFees

FROM H2HCalc;

SELECT\*FROM MarketingVPH2H;

A picture containing background pattern

Description automatically generated

\*Dummy data did not include any 2012 entries

1. Due to the nature of the server my personal SQL workbench is connected to (free community server), I am unable to connect with Python to try this out, however I believe the basis to doing this would be to use the pymysql package in Python to link to our specific query from question 4 then set up a for loop that iterates through each pair of data in the sport.csv that would look something like:

import pymysql

conn = pymysql.connect (*connection to SQL server here)*

cursor = conn.cursor()

select\_statement 1= “SELECT \* FROM FanDuelDB WHERE sport=”

part2= “AND EntryYear =”

x = [(NFL,2019),(NHL,2012),(NBA,2010),(MLB,2015),(PGA,2020)]

for a, b in x:

line= select\_statement1 + a + part2 + b

cursor.execute(line)

print(list(cursor.fetchall)

Appendix-

I will copy and paste my full SQL query here (including the creation of my dummy data tables) in case you would like to paste into your own query to check anything more in depth. I have greatly enjoyed this exercise and would appreciate any feedback you may have!

#Script Created by David Hannon

#Created for FanDuel hiring assessment

##CREATE MOCK OF ALL TABLES

#User Table- 5 Users, ensure 2 register in 2013, all make deposit

CREATE TABLE DuelUser (

user\_id VARCHAR(2) NOT NULL,

username VARCHAR(255) NOT NULL,

email VARCHAR(255) NOT NULL,

deposit\_count INT NOT NULL,

account\_suspended INT NOT NULL,

registration\_date DATE NOT NULL,

PRIMARY KEY (user\_id) );

INSERT INTO DuelUser(user\_id, username, email, deposit\_count, account\_suspended, registration\_date) VALUES ('U1','dhannon','dhannon@a.com', 5, 0, '2013-1-12');

INSERT INTO DuelUser(user\_id, username, email, deposit\_count, account\_suspended, registration\_date) VALUES ('U2','bellermeyer','bellermeyer@a.com',2,0,'2010-4-10');

INSERT INTO DuelUser(user\_id, username, email, deposit\_count, account\_suspended, registration\_date) VALUES ('U3','rboswell','rboswell@a.com',1,1,'2013-3-3');

INSERT INTO DuelUser(user\_id, username, email, deposit\_count, account\_suspended, registration\_date) VALUES ('U4','mbell','mbell@a.com',10,1,'2012-1-12');

INSERT INTO DuelUser(user\_id, username, email, deposit\_count, account\_suspended, registration\_date) VALUES ('U5','pwilt','pwilt@a.com',7,0,'2011-7-2');

#Entry Table- Ensure good portion in 2013, paid entries

CREATE TABLE DuelEntry (

entry\_id INT NOT NULL,

game\_id VARCHAR(2) NOT NULL,

user\_id VARCHAR(2) NOT NULL,

entry\_date DATE NOT NULL,

entry\_fee INT NOT NULL,

winnings INT NOT NULL,

mobile\_entry INT NOT NULL,

PRIMARY KEY (entry\_id) );

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1000,'G1','U4','2011-2-10',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1001,'G1','U3','2013-2-10',10,20,0);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1002,'G1','U5','2011-2-10',5,5,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1003,'G1','U4','2013-2-10',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1004,'G1','U1','2013-2-10',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1005,'G3','U1','2013-2-11',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1006,'G3','U3','2013-2-12',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1007,'G3','U1','2013-2-13',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1008,'G3','U4','2011-2-10',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1009,'G3','U2','2013-2-14',5,0,1);

INSERT INTO DuelEntry(entry\_id,game\_id,user\_id,entry\_date,entry\_fee,winnings,mobile\_entry) VALUES (1010,'G3','U1','2013-2-16',5,0,1);

##GAME TABLE- Ensure NFL Game

CREATE TABLE DuelGame (

game\_id VARCHAR(2) NOT NULL,

sport VARCHAR(3) NOT NULL,

size INT NOT NULL,

PRIMARY KEY (game\_id));

INSERT INTO DuelGame(game\_id, sport, size) VALUES ('G1','NFL', 500);

INSERT INTO DuelGame(game\_id, sport, size) VALUES ('G2','NBA', 200);

INSERT INTO DuelGame(game\_id, sport, size) VALUES ('G3','MLB', 2);

##Payments Table- Ensure several W and D for each user

CREATE TABLE DuelPayments (

payment\_id VARCHAR(3) NOT NULL,

user\_id VARCHAR(2) NOT NULL,

payment\_type VARCHAR(1) NOT NULL,

payment\_date DATE NOT NULL,

amount INT NOT NULL,

PRIMARY KEY (payment\_id) );

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P1','U3','D','2011-2-10',5);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P2','U4','D','2011-2-10',10);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P3','U1','D','2011-2-10',100);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P4','U2','D','2013-2-10',100);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P5','U5','D','2011-2-10',100);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P6','U3','W','2012-3-4',50);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P7','U5','W','2012-3-4',50);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P8','U1','D','2013-2-10',25);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P9','U1','D','2013-2-12',10);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P10','U2','W','2013-3-1',5);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P11','U2','D','2013-3-4',10);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P12','U4','D','2013-4-1',12);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P13','U1','W','2013-7-5',50);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P14','U2','W','2013-8-7',75);

INSERT INTO DuelPayments(payment\_id, user\_id, payment\_type, payment\_date, amount) VALUES ('P15','U4','W','2013-9-1',200);

##FILTER FOR QUESTION 2- At least one deposit and registered in 2013- Get count

SELECT COUNT(\*) FROM DuelUser WHERE deposit\_count >=1 AND Year(registration\_date)=2013;

#Question 3

#Step 1- Join All Info from Game table with the entry table

CREATE TABLE GameEntry AS

SELECT

DuelEntry.\*,

DuelGame.sport,

DuelGame.size

FROM DuelEntry

LEFT JOIN DuelGame on DuelEntry.game\_id=DuelGame.game\_id;

SELECT\*FROM GameEntry;

#Step 2- Net Deposit per user

#Adjust amount based on W or D

CREATE TABLE DuelPaymentsAdj AS

SELECT\*,

CASE WHEN payment\_type = 'W' THEN -amount ELSE amount END AS adjusted\_amount FROM DuelPayments;

#Sum NetDeposit per user

CREATE TABLE NetDeposits AS

SELECT user\_id, SUM(adjusted\_amount) AS net\_deposit

FROM DuelPaymentsAdj GROUP BY user\_id;

#Join with User table

CREATE TABLE UserPayments AS

SELECT

DuelUser.\*,

NetDeposits.net\_deposit

FROM DuelUser

LEFT JOIN NetDeposits on DuelUser.user\_id = NetDeposits.user\_id;

SELECT\*FROM UserPayments;

#Step3- Combine GameEntry and User Payment to compile all pertinent info

CREATE TABLE MasterInfo AS

SELECT

GameEntry.\*,

UserPayments.username,

UserPayments.email,

UserPayments.deposit\_count,

UserPayments.account\_suspended,

UserPayments.registration\_date,

UserPayments.net\_deposit

FROM GameEntry

LEFT JOIN UserPayments on GameEntry.user\_id=UserPayments.user\_id;

SELECT\*FROM MasterInfo;

##FILTER FOR QUESTION 3- distinct users that satisfy previous criteria plus played NFL in 2013

SELECT COUNT(distinct user\_id) FROM MasterInfo

WHERE deposit\_count >=1

AND Year(registration\_date)=2013

AND sport ='NFL'

AND YEAR(entry\_date)=2013;

##QUESTION 4##

#Step 1- Add info to MasterInfo Table regarding Fee breakdown of NFL2013 vs all other sports in 2013

CREATE TABLE MasterInfo2 AS

SELECT\*,

CASE WHEN sport = 'NFL' AND YEAR(entry\_date)=2013 THEN entry\_fee ELSE 0 END AS NFL2013\_Fees,

CASE WHEN sport != 'NFL' AND YEAR(entry\_date)=2013 THEN entry\_fee ELSE 0 END AS Other2013\_Fees

FROM MasterInfo;

SELECT\*FROM MasterInfo2;

#Step 2- Add In Mobile Entry Info per User (for Question 4)

CREATE TABLE MobileEntryInfo AS

SELECT

user\_id,

sum(CASE WHEN mobile\_entry=1 THEN 1 ELSE 0 END) AS UserMobileEntries,

count(\*) AS TotalUserEntries

FROM MasterInfo2

GROUP BY user\_id;

SELECT\*FROM MobileEntryInfo;

CREATE TABLE MasterInfo3 AS

SELECT

MasterInfo2.\*,

MobileEntryInfo.UserMobileEntries,

MobileEntryInfo.TotalUserEntries

FROM MasterInfo2

LEFT JOIN MobileEntryInfo on MasterInfo2.user\_id=MobileEntryInfo.user\_id;

SELECT\*FROM MasterInfo3;

#Step 3- Create Table of just eligible users from question 3

CREATE TABLE Eligible\_Users AS

SELECT

distinct(user\_id) FROM MasterInfo3

WHERE deposit\_count >=1

AND Year(registration\_date)=2013

AND sport ='NFL'

AND YEAR(entry\_date)=2013;

SELECT\*FROM Eligible\_Users;

##Step 4- Create table of desired info from MasterInfo3 to be joined with Eligible Users

CREATE TABLE CRM AS

SELECT

user\_id,

email,

sum(NFL2013\_Fees) AS 2013NFLFees,

sum(Other2013\_Fees)AS 2013OtherFees,

(UserMobileEntries)/(TotalUserEntries) AS PercentMobile,

MAX(CASE WHEN entry\_fee>0 THEN entry\_date ELSE 'N/A' END) AS LastPaidEntry,

net\_deposit

FROM MasterInfo3

GROUP BY user\_id, email, PercentMobile, net\_deposit;

SELECT\*FROM CRM;

CREATE TABLE CRMFinal AS

SELECT

CRM.\*

FROM Eligible\_Users

LEFT JOIN CRM on Eligible\_Users.user\_id=CRM.user\_id;

#Final Result for CRM Manager- ANSWER FOR QUESTION 4

SELECT\*FROM CRMFinal;

##QUESTION 5

#Step 1- Create table of totals of all games by year for new users (30 days)

CREATE TABLE YearTotals AS

SELECT

YEAR(entry\_date) AS EntryYear,

count(\*) AS YearTotalNewUserEntries,

sum(entry\_fee) AS YearTotalNewUserFees

FROM MasterInfo3

WHERE DATEDIFF(entry\_date, registration\_date)<30

GROUP BY EntryYear;

SELECT\*FROM YearTotals;

#Step 2- Create table of yearly Head2Head totals for new users

CREATE TABLE YearH2HTotals AS

SELECT

YEAR(entry\_date) AS EntryYear,

count(\*) AS YearTotalNewUserH2HEntries,

sum(entry\_fee) AS YearTotalNewUserH2HFees

FROM MasterInfo3

WHERE DATEDIFF(entry\_date, registration\_date)<30 AND size=2

GROUP BY EntryYear;

SELECT\*FROM YearH2HTotals;

#Step 3- Combine tables

CREATE TABLE H2HCalc AS

SELECT

YearH2HTotals.\*,

YearTotals.YearTotalNewUserEntries,

YearTotals.YearTotalNewUserFees

FROM YearH2HTotals

LEFT JOIN YearTotals on YearH2HTotals.EntryYear=YearTotals.EntryYear;

SELECT\*FROM H2HCalc;

#Step 4- Create Final Table of VP's specified output

CREATE TABLE MarketingVPH2H AS

SELECT

EntryYear,

YearTotalNewUserH2HEntries/YearTotalNewUserEntries,

YearTotalNewUserH2HFees/YearTotalNewUserFees

FROM H2HCalc;

SELECT\*FROM MarketingVPH2H;