**Interview Task**

This task is designed to check that you have some of the basic coding skills required to query and manipulate data stored in a relational database. The task is written with a view to using SQL code to answer the questions, however you can also answer the questions using Python or R code if you prefer, provided you include code for how you load the provided data into Python or R.

The data for the task is stored in the provided excel file titled *‘SQL\_Interview\_Task\_Data\_Analytics\_Engineer’*. The excel contains three sheets, each representing a table in the sample relational database. The data in these tables is a sample of customer sportsbook and casino transaction data. The three tables are:

* **customer:** a list of all customer IDs, together with the state the customer registered in
* **sportsbook\_bets:** one line per customer sportsbook bet placed, with:
  + the sport the bet was placed on
  + whether it was placed in play or pre game (in\_play\_yn)
  + the state the bet was placed in
  + the bet stake
  + the bet revenue from the company’s perspective (so negative revenue means the bet won for the customer, and they profited, and positive revenue means the bet lost and the company retained the bet stake as company revenue)
  + The date and time the bet was placed at
* **casino:** a list of all stakes placed by customers in the online casino
* **sportsbook\_totals:** one line per customer, with:
  + total\_bets = count of bets placed
  + total\_bet\_stakes = sum of bet\_stakes
  + total\_bet\_revenue = sum of bet\_revenue
  + last\_sportsbook\_bet\_placed\_date = placed date time for last sportsbook bet placed

For each of the below three questions, please provide **just the code** you would use to construct the dataset requested, treating the sheet names as table names in a database.

**Questions**:

1. For all customers in the customers table, and without using the sportsbook\_totals table, construct a dataset with one line per customer and the following customer stats:
   1. Customer ID
   2. Customer registration state
   3. Total bets placed
   4. Total bet stakes
   5. Total bet revenue
   6. Total casino stakes

SELECT cust.customer\_id,   
cust.registration\_state,   
COUNT(DISTINCT(sbets.bet\_id)) total\_bets\_placed,  
SUM(sbets.bet\_stake) total\_bet\_stakes,  
SUM(sbets.bet\_revenue) total\_bet\_stakes,  
SUM(casino.casino\_stakes) total\_casino\_stakes  
FROM "FAN\_DUAL".customers cust  
LEFT JOIN "FAN\_DUAL".sportsbook\_bets sbets ON cust.customer\_id = sbets.customer\_id  
LEFT JOIN "FAN\_DUAL".casino ON cust.customer\_id = casino.customer\_id  
GROUP BY cust.customer\_id;

1. Create a list of all customer IDs where the customer has greater than 10% of their total bet stakes placed on in play football.

SELECT outer\_query.customer\_id  
FROM(  
 SELECT inner\_query.customer\_id,   
 SUM(stotals.total\_bet\_stakes) / SUM(inner\_query.football\_bet\_stakes) as percentage  
 FROM(  
 SELECT sbets.customer\_id,   
 SUM(stotals.total\_bet\_stakes) football\_bet\_stakes  
 FROM "FAN\_DUAL".sportsbook\_bets sbets  
 LEFT JOIN "FAN\_DUAL".sportsbook\_totals stotals ON sbets.customer\_id = stotals.customer\_id  
 WHERE stotals.customer\_id IS NOT NULL  
 AND sbets.in\_play\_yn = 'Y'  
 AND sbets.sport\_name = 'football'  
 GROUP BY sbets.customer\_id  
 ) inner\_query  
 LEFT JOIN "FAN\_DUAL".sportsbook\_totals stotals ON inner\_query.customer\_id = stotals.customer\_id  
 GROUP BY inner\_query.customer\_id  
 ) outer\_query  
 WHERE outer\_query.percentage > .10  
 ORDER BY 1;

1. Find the percentage of customers who have placed bets outside of the state they registered in

SELECT (inner\_query.num\_customers::double precision) / (SELECT COUNT(customer\_id)::double precision FROM "FAN\_DUAL".customers) \* 100 AS percentage  
FROM(  
 SELECT COUNT(DISTINCT(cust.customer\_id)) AS num\_customers  
 FROM "FAN\_DUAL".customers cust  
 LEFT JOIN "FAN\_DUAL".sportsbook\_bets sbets ON cust.customer\_id = sbets.customer\_id  
 WHERE cust.registration\_state <> sbets.bet\_state  
 ) inner\_query;

1. For all customers in the customers table, construct a dataset with one line per customer and the following customer stats:
   1. Customer ID
   2. A list of all states where the customer has placed a bet

import pandas as pd  
from dataclasses import dataclass  
  
  
def main():  
 data = Data()  
 print(data.get\_result\_dataframe())  
  
  
@dataclass  
class Data:  
 customer\_data\_file\_path = '~/Documents/FanDual\_Technical\_Interview\_Materials/csv\_files/customer.csv'  
 sportsbook\_bets\_file\_path = '~/Documents/FanDual\_Technical\_Interview\_Materials/csv\_files/sportsbook\_bets.csv'  
  
 def get\_result\_dataframe(self):  
 joined\_df = self. get\_joined\_dataframe()  
 joined\_df = joined\_df.dropna()  
 joined\_df['concat\_column'] = joined\_df['customer\_id'].astype(str) + ' ' + joined\_df['bet\_state']  
  
 list\_obj = joined\_df['concat\_column'].unique().tolist()  
  
 result\_df = pd.DataFrame(list\_obj)  
 result\_df = result\_df[0].str.split(' ', expand=True)  
 result\_df.rename(columns={0: 'customer\_id', 1: 'bet\_state'}, inplace=True)  
 result\_df = result\_df.groupby('customer\_id')['bet\_state'].apply(list).reset\_index()  
 result\_df['customer\_id'] = result\_df.index  
  
 return result\_df  
  
 def get\_joined\_dataframe(self):  
 joined\_dataframe = self.get\_customer\_data().merge(self.get\_sportsbook\_bets\_data(), on='customer\_id', how='left')  
 return joined\_dataframe[['customer\_id', 'bet\_state']]  
  
 def get\_customer\_data(self):  
 customer\_dataframe = pd.read\_csv(self.customer\_data\_file\_path)  
 return customer\_dataframe  
  
 def get\_sportsbook\_bets\_data(self):  
 sports\_book\_bets\_dataframe = pd.read\_csv(self.sportsbook\_bets\_file\_path)  
 return sports\_book\_bets\_dataframe  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

1. Using data in the sportsbook\_bets table, write code to update customer stats in the sportsbook\_totals table, fulfilling the following requirements:
   1. Customer stats in sportsbook\_totals table are updated to reflect any additional bets placed by a customer after a customer’s last\_sportsbook\_bet\_placed\_date in the sportsbook\_totals table
   2. For customers with bets in the sportsbook\_bets table, but no record in the sportsbook\_totals table, add a new record in the sportsbook\_totals table for the given customer(s) with their relevant stats

INSERT INTO "FAN\_DUAL".sportsbook\_totals (customer\_id,   
 total\_bets,   
 total\_bet\_stakes,   
 total\_bet\_revenue,   
 last\_sportsbook\_bet\_placed\_date)  
SELECT sbets.customer\_id,   
COUNT(sbets.bet\_id) AS total\_bets,   
SUM(sbets.bet\_stake) AS total\_bet\_stakes,   
SUM(sbets.bet\_revenue) AS total\_bet\_revenue,   
MAX(sbets.bet\_placed\_date) AS last\_sportsbook\_bet\_placed\_date  
FROM "FAN\_DUAL".sportsbook\_bets sbets  
WHERE NOT EXISTS (SELECT   
 FROM "FAN\_DUAL".sportsbook\_totals stotals  
 WHERE sbets.customer\_id = stotals.customer\_id  
 )  
GROUP BY sbets.customer\_id;