**Example Exam 2 – Q1 Solution**

**Question 1, a**

select

gs.country

from

gasstations as gs

where

gs.GasStationID = 7;

**Question 1, b**

select

avg(t.price)

from

transactions as t;

**Question 1, c**

select

c.CustomerID

from

customers as c

left join

transactions as t

on

c.CustomerID = t.CustomerID

where

t.CustomerID is null;

**Question 1, d**

create view segmentMismatch as

select

t.CustomerID,

gs.GasStationID,

c.Segment as CustomerSegment,

gs.Segment as GasStationSegemnt

from

transactions as t

join

customers as c

on

t.CustomerID = c.CustomerID

join

gasstations as gs

on

t.GasStationID = gs.GasStationID

where

c.Segment != gs.Segment;

**Question 1, e**

I used the segmentMismatch view I created in Question 1(d).

The use case I thought of is calculating the total amount of money a customer spent at gas stations belonging to a different segment.

select

CustomerID,

SUM(price) as TotalMoneySpent

from

segmentMismatch

group by

CustomerID;

**Question 1, f**

Since no explicit transactionID I decided to combine 4 fields to act as the ID, Date, Time, CustomerID and GasStationID. I assume this combination will be enough to act as a unique key for the table.

create view gasstation\_trans\_count as

select

gs.GasStationID,

count(distinct concat(t.date, '\_', t.time, ‘\_’, t.CustomerID, ‘\_’, t.GasStationID)) as numOfTransactions

from

gasstations as gs

left join

transactions as t

on

t.GasStationID = gs.GasStationID

group by

gs.GasStationID;

**Question 1, g**

I used the gasstation\_trans\_count view I created in Question 1(f).

select

numOfTransactions,

count(GasStationID) as numOfGasStations

from

gasstation\_trans\_count

where

numOfTransactions is not null

group by

numOfTransactions

order by

numOfTransactions;

**Question 1, h**

I used the gasstation\_trans\_count view I created in Question 1(f).

select

numOfTransactions,

count(GasStationID) as numOfGasStations

from

gasstation\_trans\_count

group by

numOfTransactions

order by

numOfTransactions;

**Question 1, i**

select

t1.CustomerID,

t1.GasStationID,

t1.Date,

t1.time as Trans1Time,

t2.time as Trans2Time

from

transactions as t1

join

transactions as t2

on

t1.CustomerID = t2.CustomerID

and

t1.GasStationID = t2.GasStationID

and

t1.Date = t2.Date

and

t2.time > t1.time;

**This query is asymmetric**

**Question 1, j**

In my opinion, there is no reason to retrieve both (a,b) and (b,a), and this is also reflected in my query.

If our goal is to detect errors or fraud, it’s enough to receive information on a suspicious pair of transactions once.

Repeating the same pair in reverse order doesn’t provide any additional value for determining whether the pair is problematic.