Data Scientist – technical test

These questions are designed to test your logical thinking, data modelling, stats and

programming ability.

Some tips:

- - There is no time limit to finish the test but we would like to receive your answer

  during a week since you receive the technical test (Some guidance is given below about how long we expect you to take on each question, but this will vary a lot depending on your skillset)

- - An important skill for data scientist is to validate your work, so please check your answers (and where appropriate show how you have checked them)

  Once you have completed your test, please email the answer (you can type out answers on a word document or github repo).

Part 1. Logic (10-20 minutes)

A cube is painted Blue on all six sides. It is divided into 125 (=5 x 5 x 5) equal smaller cubes.

Find:

1. The number of smaller cubes having
   1. a) 3 faces coloured?

      *Those are corners of a cube. Cube has **eight (8)** corners.*

   2. b) Exactly 2 faces coloured?

      *Those are edges of the cube except for corners. (N-2)*12 = (5-2)*12 = **36***

   3. c) Exactly 1 face coloured?

      *Those are insides of the cube faces. ((N-2)^2)*6 = ((5-2)^2)*6 = **54.** Or it is number of all painted cubed minus results above, that is: 5*5*2 (top and bottom faces)+3*5*2(front and back faces excluding top and bottom faces)+3*3*2(side faces exluding top, bottom, front, and back faces)-8-36=50+30+18-8-36=54 (results match)*

   4. d) 0 faces coloured?

      *Those are insides of a cube. It is a cube with edge N-2. (N-2)^3 = (5-2)^3 = **27.** Or it is number of all cubes minus painted cubed. 125 – 8 – 36 – 54 = 27 (results match)*

2. All 125 cubes are put into a bag. If a single cube is selected at random from the bag, find probability of picking a cube having 1 or more Blue faces

*There are [(N^3=125) – 4.d] small cubes with at least one face painted. That is 125-27=98. So the probability is **98/125=0.784***

3. What is the average number of Blue faces on a small cube?

*There are eight (8) small cubes with three (3) faces coloured, 36 small cubes with two (2) faces coloured, 54 small cubes with one face coloured, and 27 with zero (0) faces coloured. So the average is (8*3+36*2+54*1+0*27)/(125)=(24+72+54)/125=150/125=1.2. However one may (and most probably should) go for "among all cubes how many faces has the most common cube", which stands for the mode and is equal to one (1); or go for "among all cubes what is the number of faces coloured that 50% of all the cubes do not exceed", which stands for median and is also equal to one (1)*

In the above situation N=5, (with N^3 =125)

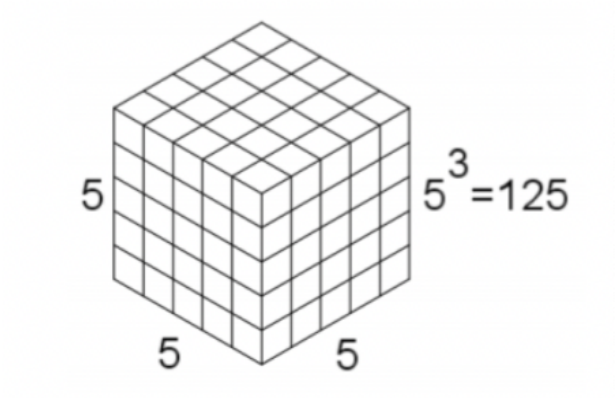
4. For general N, give a formula for (1.2) the number with exactly 2 faces coloured

Those are edges of the cube except for corners. (Edge length – tail and head of the edge)#_of_edges = **(N-2)*12.***

5. For what values of N is this formula correct?

This formula is correct for N>=2, for N=1 the answer is zero (0).

***GERERAL NOTE:** illustration below is quite good and can be used for self check: one can simply count "with fingers" with the help of it. Also in order to check oneself it is a good idea to answer question independently and then check if the answers are in line.*



Part 2. SQL (15-30 minutes)

We have the following tables:
user - table with information about registered students

Column name	Datetype	Description

id	integer	Unique id of user
date_joined	timestamp	Date and time of registration
country_code	varchar(2)	country of user (2-letter country abbreviation)

payment - table with information about payments

Column name	Datetype	Description
id	integer	Unique id of payment ((one student can have from 0 to X payments)
user_id	integer	Id from table user
payment_amount	float	Paid amount in USD
created_at	timestamp	Date and time of the payment

lesson - table with list of lessons. Possible statuses of lessons:

CONFIRMED - lesson happened successfully *SCHEDULED* - for future lessons *CANCELED* - for lessons that were canceled

Each lesson appears on the table only one time. Lesson status is being updated.

Column name	Datetype	Description
id	integer	Unique id of lesson ((one student can have from 0 to X lessons)
user_id	integer	Id from table user
status	varchar(255)	Current status of lesson (can be updated)
created_at	timestamp	Date and time of the payment
hours	integer	Duration in terms of hours of the lesson

Write SQL query that returns:

1. number of registered users by country

```
select country_code,
       count(*)
  from user
 group by country_code
```

2. % of users, who made their first payment in 3 days after registration by country

```
select u.country_code,
       count(distinct p.user_id)/count(distinct u.id)
  from user u left join payment p on u.id = p.user_id
                    and days(p.created_at – u.date_joined)<=3
 group by u.country_code
```

3. % of users, who made their first payment in 3 days after registration and had 2 confirmed lessons in 7 days after registration by country

Subqueries are going to be used. We will go for the following syntax of the aliases: WITH alias1 AS (subquery1)… aliasN AS (subqueryN). This is done in order to make code easy to read and track. This syntax is different for different SQL engines and often is not allowed for users with few privileges. In that case one should put subqueries in "()" brackets into the general query.

```
WITH
three_days_reg AS
(
select distinct u.id
```

```
  from user u inner join payment p on u.id = p.user_id
                                and days(p.created_at – u.date_joined)<=3
)
two_lessons_week AS
(
select u.id
  from user u inner join lesson l on u.id = l.user_id
                                and days(l.created_at – u.date_joined)<=7
                                and l.status = 'CONFIRMED'
 group by u.id
having count(*)>=2
)
select u.country_code,
       count(distinct t.id)/count(distinct u.id)
  from user u
       left join
       (
                select tdg.id
                  from three_days_reg tdg inner join two_lessons_week tlw on tdg.id=tlw.id
                ) t
       on u.id = t.id
 group by u.country_code
```

4. % of weekly new users that never have done a payment

```
select count(distinct (case when p.id is NULL then u.id end))/count(distinct u.id)
  from user u left join payment p on u.id = p.user_id
 where u.created_at >= trunc(sysdate)-7
```

5. Advanced level (Extra point): Write the SQL that returns how many hours of confirmed
   lessons a specific user (for example user_id=1) has taken between payments.

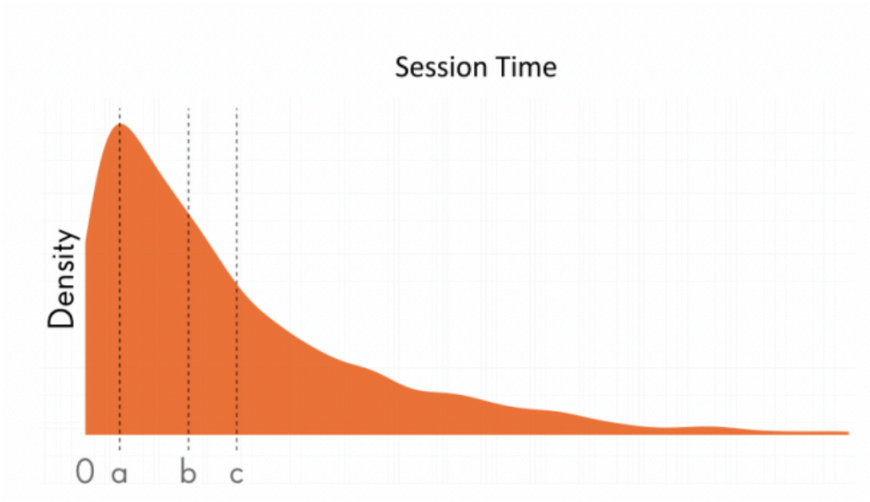
```
-- Calculation w/o taking into account that lessons may overlap with payments
WITH user_pauments_min_max AS
(
select user_id,
       min(created_at) first_payment,
       max(created_at) last_payment
  from payment
 group by user_id
)
select u.user_id,
       sum(l.hours)
  from user_pauments_min_max u inner join lesson l on u.user_id = l.user_id
                                          and l.status = 'CONFIRMED'
                                          and l.created_at > u.first_payment
                                          and l.created_at < u.last_payment
 where u.user_id = some_specific_user_id
 group by user_id

-- Calculation with taking into account that lessons may overlap with payments
WITH user_pauments_min_max AS
(
select user_id,
       min(created_at) first_payment,
       max(created_at) last_payment
  from payment
 group by user_id
```

```
)
select u.user_id,
      sum(case when (l.created_at > u.first_payment)
                        and (l.created_at+l.hours/24 < u.last_payment) then l.hours
                  when (l.created_at < u.first_payment)
                        and (l.created_at+l.hours/24 < u.last_payment) then l.hours -
hours(u.first_payment-l.created_at)
                  when (l.created_at > u.first_payment)
                        and (l.created_at+l.hours/24 > u.last_payment) then
hours(u.last_payment-l.created_at)
                  when (l.created_at < u.first_payment)
                        and (l.created_at+l.hours/24 > u.last_payment) then
hours(u.last_payment-u.first_payment)
                end)
  from user_pauments_min_max u inner join lesson l on u.user_id = l.user_id
                                          and l.status = 'CONFIRMED'
                                          and l.created_at+l.hours/24 > u.first_payment
                                          and l.created_at < u.last_payment
 where u.user_id = some_specific_user_id
 group by user_id
 /*
 one may check "case" statement by using the following picture:
------------------------
 -----  -------   -------
***P*****************P***
where "*" stand for the timeline with "P" as payment
"-" stands for lesson cases duration,
each character is a single step in the timeline
Note: picture does not look good but is informative in basic text editor
 */
```

GENERAL NOTE: How does one check SQL queries? First, keep it simple even with the cost of computation efficiency, then, if efficiency is critical, one should optimize query and compare the results; one may do it on a sub slice of a data by partition. Second, use some different tool from SQL technology, that you are comfortable with, to compare results. It can be Excel, R, python. Third, be aware of NULLs; usually weird errors come from NULLs in joins, aggregation functions, where statements. Forth, check for obvious mistakes like shares greater than 100%. Fifth, test query on artificial made-up data on which it is easy to verify results. Sixth, do a case study. For example, if results are per user, get a single user, calculate agnostically to query its results and track the dataflow for that user. And last but not least, compare results with existing report or BI tools like Tableau. MicroStrategy, Looker; results one gets should be generally in line with preexisting results.

Part 3. Statistics (5-10 minutes)

!



1. What standard measures are a,b and c most likely denoting? Please explain why.

One may assume that the chart represents smoothed frequency chart of session lengths which looks like lognormal distribution density chart (and is natural to assume to be lognormal). "a" stands for mode of distribution, which is the maximum value that PDF of distribution reaches. For discrete distributions mode is the value that occurs most often, for continuous distributions (lesson time or session time may be assumed as continuous at some extent) point of mode may be treated as center of an interval of values that occur most often. "b" and "c" are (most likely) median and mean values. Median point splits chart by two figures with equal areas ("b" looks like such point). Mean values ("c") is shifted to the right due to "long fat" tail of the distribution (which is also natural to assume as there might be some significant amount of 12h+ sessions comparing to probably ~1h mode session)

2. If you drew n samples from this distribution and measured their mean, then repeated that many times, how would you expect the distribution of those sample means to differ from the distribution?

According to central limit theorem (and many existing computer experiments) one should expect distribution to be similar to N(mu, (s^2)/n), where "N" stands for normal distribution, "mu" is a mean value of described distribution, and "s^2" is variance of described distribution. The larger the "n" the more normal-like distribution one should expect. As it was stated earlier one may assume that described distribution is lognormal. So, long story short, described distribution is lognormal-like, distribution of mean described in 2. is normal-like with the same mean and variance lower in "n" times.

3. Would its standard deviation be bigger, smaller or the same as this distribution's standard deviation and why?

As it was stated in 2. variance is expected to be lower and equal to (s^2)/n according to central limit theorem, where s^2 is variance of distribution described by chart. So, (standard) deviation is going to be lower and equal to s/(n^0.5)

Part 4. Storytelling (10-20 minutes)

Describe a technological abstract concept of your choice (for example: internet, electricity, credit card, email, slack, ...) to a 4 years old child. Please make sure you don't use other complex concepts while describing it and use a plain language in order to maximize the

chances that the child will be able to understand it. We are going to evaluate your capacity to simplify and explain complex subjects.

(Mínimum 100 words, maximum 400 words)

Let's go with internet.

Internet is a thing that stores almost all the things people know, all the books, movies, music, and pictures. It is like a shelf with lots of stuff. If you have a computer or a phone that can use internet, you can ask a question and your computer (or phone) will go to that shelf and look for an answer. You can ask to show you a picture, or a cartoon, or play a song. If you draw a picture or made a photo or a video you may put it into internet. Just as you can do with a shelf but with a phone or a computer. That is how other people may see it. Or you can hide it on a shelf, and only you and people you choose can see it. You can even talk and play with your friends with internet. Just use your computer or phone to write them a letter or make a video. But be aware of bad people that may trick you with internet. They can pretend that they are your parents or friends.

Internet is not a single place. All the things people put into internet are on many shelves all over the world. They are called servers. Computers and phones can go there and look for the things you asked for.

And that is what internet is.

Part 5. Recommendation Engine (+60 minutes)

Preply wants to include a small recommendation engine to suggest potential good tutors based on a collaborative filtering. We want to show "users that viewed this tutor also viewed..." section within the tutors profile page.

Given a Preply profile views dataset, make a small recommendation system that given one profile_id returns the 10 most potential tutors to be viewed based on a collaborative filtering. We want to consider potential good tutors those tutors that have been viewed by users that also viewed our input tutor_id. The return should include the top 10 potential tutor_id's (ordered by popularity).

The dataset includes the following columns:
- id: sequential id of the profile views table (int)
- user_id: unique code that identifies the users viewing the page (varchar(255)) - tutor_id: unique code that identifies our tutors (varchar(255))

Each row of the dataset represents one tutor profile view. A tutor profile view is triggered when a user visits a tutor profile page to view more information about their experience teaching.

Dataset: https://docs.google.com/spreadsheets/d/e/ 2PACX-1vTyQAs46R3D8qJcvrXhl0E03fEGb8kN1kMLx1KNA_HbAwBcHOOx_eJBXU8pnplPOTdQgiQOsDEBzX5L/pub?gid=1102523268&single=true&output=csv

Questions:

1. Which tutors will your recommendation engine return given the tutor_id “ff0d3fb21c00bc33f71187a2beec389e9eff5332”? Will it work for any tutor_id of the dataset?
   Recommendations for ff0d3fb21c00bc33f71187a2beec389e9eff5332 are
   ['e9ee460fac3c729a7de68f933621a117878dad2d',
   '0d3dc58ead1aa17dcc7d6481215d0e940f1cedad',
   '2b7f6844166a1543562f4cf17c9bca319db1c3bf',
   'c1cb5cef4e3d6cdb91e677a74d2aeec64da15277',
   'bba19f79a4d8823fd39401b23c31df91e4c5bb74',
   '8291ae14f3d7f882e6d258e4e263664fc9b98e0d',
   '709a505d929fcf4e3fc291e8d5cdc713d52c200b',
   'e1a9933aa1834b272ffcad6d62577bb4fdc196ea',
   'ce728286e79668922b5e461b62aab7a6fa3dd616',
   '19e13adb133a06778dfacc16731c56408d8a128e']

   Here is a catch with this tutor. Starting from place eight (8) to place 11, most popular users have the same ratings. So in order not to give an advantage to some tutors that appear on top because they have ids that system puts on top (for example, sort them in alphabetic order), one should choose random users for vacant places of the grid.
   The recommendation engine that I built does it. It also fixes the random state of this random draw, so the result should be replicable, which is critical for production incidents investigations and scientific studies.

   Top 15 users for tutor_id:
   tutor_id
   e9ee460fac3c729a7de68f933621a117878dad2d 39.0
   0d3dc58ead1aa17dcc7d6481215d0e940f1cedad 37.0
   2b7f6844166a1543562f4cf17c9bca319db1c3bf 34.0
   c1cb5cef4e3d6cdb91e677a74d2aeec64da15277 33.0
   bba19f79a4d8823fd39401b23c31df91e4c5bb74 32.0
   8291ae14f3d7f882e6d258e4e263664fc9b98e0d 31.0
   709a505d929fcf4e3fc291e8d5cdc713d52c200b 30.0
   ce728286e79668922b5e461b62aab7a6fa3dd616 29.0
   19e13adb133a06778dfacc16731c56408d8a128e 29.0
   e1a9933aa1834b272ffcad6d62577bb4fdc196ea 29.0
   b4be83829f480ee08cb4f462f3e3c5050517400a 29.0
   8f7079ff8a3586bd66298b2095e9db74bca7201b 28.0
   8ed927ca41449c7da3518b08e63c8d6fe7c8411f 27.0
   8e5ee419cbe2a3589e012a65ae8b97068414f336 27.0
   d22517e2897752a190955ad862d2dec43248e83e 26.0

   The recommendation engine will work for any tutor from the dataset formally. However, results may be questioned. In case that tutor has few relevant tutors, recommendation may have poor quality. I will describe this situation in detail in question 3.

2. What you will suggest to do to avoid a cold start of a tutor (a tutor that has not yet received any view)?
   Tutors that have been a long time with Preply have a significant advantage. An ultimate disadvantage is a brand new user with zero views. To keep the ecosystem of service healthy, one may give recommendations based on tutors split by cohorts of how long they had been with the Preply.
   One may also use data other than users’ activity to improve "cold start" and cohort recommendations. This data may be interests, teaching experience, geolocation or

other relevant features from a business point of view and general wisdom of service specifics gathered by service.

3. What will you do if given a tutor_id the recommendation system returns less than 10 potential tutor_ids?
   One way is to show less than 10 recommendations. It makes sense if we want to show exactly “users that viewed this tutor also viewed...” and anything other might be treated by the user as annoying. But I went for another approach. I case of less than 10 available popularity recommendations, I fill vacant places with “based on the tutor you visited you may also like this” recommendations.
   When there are less than 10 available recommendations, I fill vacant places with tutors I choose from matrix factorization. Here is how it works: I estimate user-tutor interaction matrix (let it be A) as a product of two matrixes – A=U*T, where dimension of A is [#_of_users, #_of_tutors], U is [#_of_users, #_of_latent_variables], T is [#_of_latent_variables, #_of_tutors]. #_of_latent_variables is a number of some latent unseen from matrix A features that describe both users and tutors. Then I choose tutors most similar to tutor_id from matrix T (I used cosine similarity in between tutors vectors)
   The same technique is used in factor analysis (https://en.wikipedia.org/wiki/Factor_analysis). Let's illustrate it with example. Let's say that N students take M tests, and we record their performance in each test. All the students have a talent for maths, languages, and arts. Tests are, on the other hand, have a different share of tasks for math, languages, and arts. Here #_of_latent_variables is three (3), and those latent variables are math, languages, and arts measurements. By knowing a student's characteristics and test characteristics, one can predict how student will perform.
   Now in our case, we don’t know #_of_latent_variables and their meaning. Nevertheless, we can estimate the user-tutor interaction matrix. I tuned this parameter that estimation is at least 80% accurate (not 100% as in fact, users may have a bad experience, so I don’t want to mimic matrix A exactly). Why 80%? This is a number that I cannot prove to be right with provided data, but it served me well before, so I used it as a rule of thumb. If one wants to use it in production, some experiments and validations with previous results and knowledge should be made in order to find perfect accuracy.
   Returning to question 1. In case there are few relevant tutors for popularity recommendation, vacant places will be filled with matrix factorization results, which might be inaccurate for this specific case, as there are few similarities with other users. So overall recommendation might be inaccurate and quite bad.
4. If the dataset was including the possibility of a user to view a tutor_id more than once, how it will influence the recommendation system? Which dataset do you think will perform better and why?
   Tutors with multiple visits from a user would have been treated as more popular. For example, a tutor with two visits from the same user, and with visits only from that user, would have been treated as twice popular as compared to “one user -- one record” approach. Tutors' similarities from matrix factorization would also change in a way that may hurt recommendations.
   I don’t think that with that kind of data recommendation system would perform well. Because multiple visits from the same users are not exactly similar to the rating of satisfaction. One may argue that sometimes it might be the opposite, like if that was a perfect user-tutor match, then why visit page twice. Also, multiple visits relate to technical issues with the user’s browser, doubts of the user, the user’s free time, etc. Again, multiple visits from a single user should not be treated as ratings and are not relevant; thus, it should be treated as noise in data.
5. Try to reason why do you agree or disagree on removing from the dataset, the rows where tutors are visiting their own tutors page?

I believe that those rows should be excluded. The main goal of the recommendation system is to make recommendations as satisfactory for the user as possible and eventually help to find the right tutor. And tutor that visits his own page is not looking for classes, so that record is irrelevant.
Now, with the same logic, one may ask if we should exclude all tutors that browse tutors' pages. The answer is “yes” if we can confirm that they are not looking for classes but rather look at other tutors, the answer is “no” if the tutor does look for classes (we can confirm it for example if the tutor eventually took a course of lessons), then it should be treated as regular user’s visit.

6. Do you have any improving proposition that could maximize Preply’s tutors page views? (different algorithm? Different dataset? ...)

- First of all, it would be great if the users-tutor matrix would not be sparse binary but contained ratings (for example, the number of hours of lessons eventually taken). This would give so much room for improvement both in the current approach and by opening new techniques like “slope one” recommendations. And it would allow us to perform offline validation of the recommendation system.
- Data gathered from users as questioneer like “why would you like to study?”, “do you already have some experience?”, “do you consider yourself an introvert?” (some of this data and more is already gathered by Preply from my experience) would also improve recommendations. Not only it may solve the cold start problem, but it also can give its own recommendations that may be put in cooperation with collaborative filtering. It is well known that bets modern recommendation systems are hybrid like that.
- It might be a good idea to ask for a review from a user who took a class. This data can be used for recommendation and recommendation performance validation. And with modern Natural Language Processing techniques is it way easier than one might think.
- One should check already made recommender engines from Spark, Amazon, Google, Microsoft, and others. It is unwise not to try them out at least to see how well the current custom system is doing in comparison. Library Surprise (http://surpriselib.com) might also be considered as well as others already made engines.
- The technique described in question 3 as “based on a tutor you visited, you may also like this” is worth checking. It does suffer from scalability, sparsity, the volatility of predictions, and other problems, but having questioner data, one can try to perform a study of the meaning of latent features, then predict them and then try to solve cold start problem and make a good meaningful recommendation.
- As was stated earlier, new tutors should be given a chance, and it is a good idea to force them into recommendation one way or another.
- It is a good idea to make a distinct recommender system for distinct cohorts of users and tutors based on the time they have been with the Preply, as we can be more sure of what is old user/tutor is like than a new one. Plus, they most definitely have different behavior and new users have less views and lessons than old ones.
- One should keep the ecosystem of recommendations healthy, keep discoverability of recommendation high enough. If service is stuck in a loop of “make recommendations based on users experience -> users use only what we recommend them,” it is not good. So forcing something different from original recommendation into recommendation grid (like recommendations from user data unrelated to service usage).
- Using deep learning in the recommendation engine is very promising. It may provide stable, always relevant, continually improving solution. But it is challenging, and like all complex black-box models, should be treated with caution.
- One should use anomaly detection and seasonality analysis in order to improve data quality and exclude anomalies from training samples. A good start is fbprophet library (https://facebook.github.io/prophet/).