THE TRACK FITNESS GYM CASE MEMO

Roll Number: 21f2000426

Name: Arnav Kumar

Links: All necessary calculations, ipynb files, and excel files are in the link below:

https://drive.google.com/drive/folders/1i55sTYUMwu-7IabznuEpIcSpXup8nhuR?usp=share_link

Executive Summary:

A further dive into the data generated by the track fitness club over the period of 8 months Unlatched many interesting discoveries. The track Fitness studio is one of the best gyms in the Nagpur district with several facilities like a thermal spa, a well-planned parking area with security cameras, more than enough equipment to train every muscle group, and so on. Gym culture in my hometown is fairly new since most gyms that existed / still exists before the track fitness club offered minimal to no facilities with no personal training support and sub-optimal facilities for women because of which most people refrained from ever getting a gym subscription, but the track fitness club changed everything.

Regardless of everything the new fitness studio offered in a fairly new environment it suffers from two major issues which were discussed frequently with the responsible staff and the owner, the elephant in the room we are talking about here is **poor client retention rates** and **overcrowding at peak hours** the direct outcome of which as per the owner is unsteady and subpar revenue which is the primary problem I was tasked to tackle with

A quick fix to **the overcrowding** enigma will help the clients to secure gym equipment without having to wait for longer periods which can be awkward and is one of the practical reasons a client might leave the establishment to never join again, once this problem is taken care of clients are more likely to continue with their subscriptions periodically i.e., they can be retained for longer period of times which is a win for the business.

Both the problem statements I have been working on are beautifully intertwined with each other which takes me to the next issue which is **Client Retention**. When and if the retention is subpar a business suffers unpredictably, one month you are looking at 70 new clients, and the other month new registrants might be as low as 15, Unpredictability of the revenue in this business is the stepping stone towards an epic disaster and might even lead to losses in future.

To help the business with its ordeal I collected the **invoice bills**, punch-in punch-out (**PIPO**) digital logs and client personally Identifiable Information (**PII**) all the data collected reflects the day-level operations of the fitness studio for a total of 240 days approximately from January 1, 2022, to august 31st 2022

After preprocessing the data, using all the tools at my disposal, and given the time constraint I have come up with the solution to the best of my abilities, tools I have used mainly consist of Microsoft excel, jupyter notebook as an IDE, and python as my main programming language including some popular packages like statistics and pandas helped me a lot.

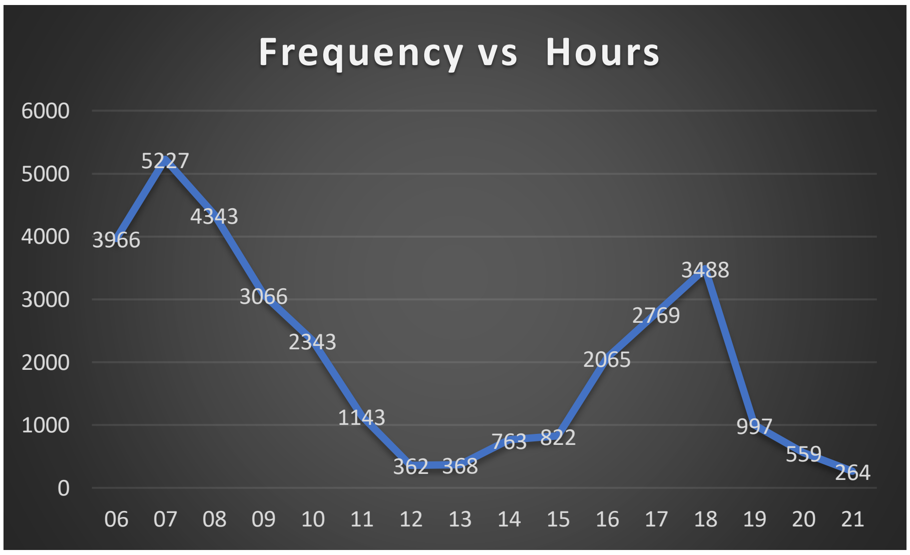
Detailed Explanation of Analysis Process/Method:

Out of the two paramount adversities, I was dealt with I decided to tackle the Overcrowding issue first doing this later aided me to simplify the analysis and breakdown of the second problem statement which is poor client retention rate.

1. Overcrowding:

To inspect and visualize the overcrowding at peak hours, I tidied up the data which involved some meticulous cleaning, and extracted the hour from the generated timestamp of the format (yyyy-mm-dd hh.mm.ss) and plotted a Pivot table, the graphical visualization of that pivot table gave me a crystal-clear idea about the hours which were the primary victims of client congestion

Below is the pictorial representation of the same:



It is clearly evident that 6 to 8 am and 5 to 7 pm are prime time intervals of traffic jams in the gym with 7 am and 6 pm being the worst of all as indicated by the sharp peaks, the ascent and descent before and after the peaks respectively is the steepest at the said hours, which exactly means that these hours every day for the history of 8 months was the time of major inconvenience for the clients which they had to put up with primarily because **their workout**

schedules are not flexible that means they have other commitments which they have to prioritize and only after those commitments are addressed they show up in the gym, the problem is most of them show up in the at the same time which causes such sharp spikes or peaks seen at 7 am and 6 pm.

To address this issue, I set on the quest to **find the tiny percentage** of clients who are actually flexible with their workout schedule but prefer to hit the gym at the peak hours, which results in overcrowding, the **discovery of such flexible clients** and limiting these clients to such a slot which is away from the peak hours will surely help the business, various techniques like offering some incentives like special discounts for this clients to visit a particular slot can be put in place to avoid congestion which can be possibly avoided

I followed a **4-step plan of action** to find such hidden robin hoods which though exist in a minority but beyond any doubt do exist.

1. Step 1 includes **categorizing the clients into their most preferred (mode) slots** meaning, I needed to find such slots in which the client has trained the most.

2. After client categorization, I **determined the % deviation of the client from their preferred slot**

3. And to finally seal the deal, the identification of flexible clients was carried out by **grouping all such clients together who had deviated more than 25 percent of the time** throughout their workout frequency (adherence rate)

4. **Identification of special flexible clients**.

   Each step is explained in a detailed manner as follows

- **STEP 1. categorizing the clients into their most preferred slots:**

To achieve the current objective, a **list of all the unique clients** was extracted from the invoice dataset this list of unique clients is a list of 494 unique registration numbers the gym witnessed in its entirety of 8 months, **client_unique** = [943,953, 1018….]

necessary variables which defined the slots were determined as follows

morning_slot = [6,7,8,9,10]

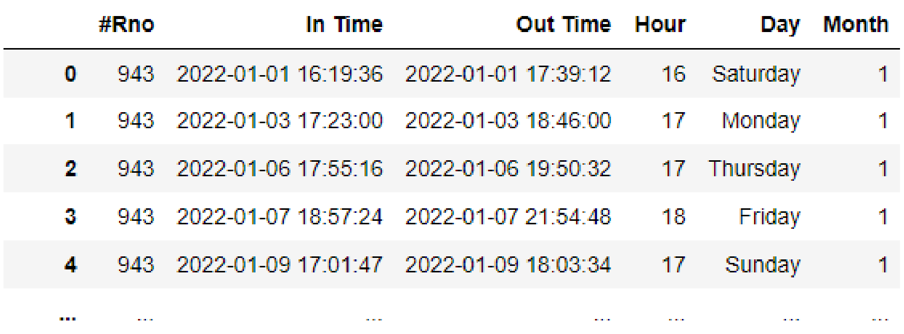
noon_slot = [11,12,13,14,15]

evening_slot = [16,17]

late_evening_slot = [18,19,20,21]

where each variable above is a list datatype and the value inside the lists are of numeric datatypes, the elements inside the list corresponds to the hour (in 24 hours format)

A. I imported the invoice dataset and PIPO dataset into a DataFrame via the pandas library into my working python environment, then I created a python dictionary which is expected to hold the key-value pairs, **where a key corresponds to the unique Registration number of the client and the value corresponds to a list of all the slots in which the client had entered the gym for his or her workout session**, below is an easy example of what this dictionary holds, I have also linked an **image of the PIPO dataset** below for easy reference



	#Rno	In Time	Out Time	Hour	Day	Month
0	943	2022-01-01 16:19:36	2022-01-01 17:39:12	16	Saturday	1
1	943	2022-01-03 17:23:00	2022-01-03 18:46:00	17	Monday	1
2	943	2022-01-06 17:55:16	2022-01-06 19:50:32	17	Thursday	1
3	943	2022-01-07 18:57:24	2022-01-07 21:54:48	18	Friday	1
4	943	2022-01-09 17:01:47	2022-01-09 18:03:34	17	Sunday	1
...	...	...	...	...	...	...

The **above DataFrame labeled as** "**df_pipo**" is traced via for loop and a **dictionary labeled as "client_slot"** is assembled from scratch. The dictionary is of the following format:

{#Rno: list, #Rno: list, ……...}

The client_slot dictionary for the first 5 visible observations (in the above image) will appear as follows:

❖ **client_slot_dict = {943: ["evening slot", "evening slot", "evening slot", "late evening slot", "evening slot…], 953: [……], 1018: [……], …}**

The dictionary was created with the help of a unique client list and PIPO DataFrame, notice that instead of appending the corresponding hours as it is to the list, I have taken a step forward and appended the corresponding slot (in text datatype) instead, for example **in the above image** we can see that **for the first observation**, the Hour column corresponds to '16', and **16 corresponds to the evening_slot** as per our slot initialization, hence **"evening slot"(of text datatype)** will be appended to the list as an element, not the Hour itself. The following process is carried out iteratively for all the unique clients.

B. After the above mini accomplishment where we now have the information of the count of each and every slot, each and every client visited in the history of his or her subscription arranged sophistically in the dictionary, we can finally proceed to gather more information about the most reoccurring slot for every client, we can also say the most reoccurring slot in every list technically speaking, **the mode element of the list** of the **client_slot_dict** is the preferred slot of the client.

I set out to extract this viable piece of information by first importing the **statistic package** from the python library, the statistic package has an inbuilt function called **mode,** which returns the maximum occurring element from a given list of elements, for example:

mode ([“evening slot’, “noon slot”, “noon slot”, “noon slot,”, “morning slot”]) will return a “noon slot” as this element is the most reoccurring element in the list and therefore

mode ([“evening slot’, “noon slot”, “noon slot”, “noon slot,”, “morning slot”]) = “noon slot”

The above logic will be carried out iteratively for each and every key in the **client_slot** dictionary which will give birth to a new dictionary called **client_preferred_slot_dict**. The newly acquired dictionary **will have a unique registration number as the key** and **a preferred slot as the value** of those keys, the **client_preferred_slot_dict** will have the following format:

client_preferred_slot_dict = {#Rno: “preferred slot of this #Rno”, #Rno: ……., #Rno: …}

in actuality the **client_preferred_slot_dict** will appear as follows:

- **client_preferred_slot_dict = {#Rno: “noon slot”, #Rno: “morning slot”, ….}**

Now we have the particulars about the preferred slots of each and every 494 unique clients and this is how I have categorized every client into specific slots, having the information stored in a dictionary.

STEP 2. Determination of deviation (in %) by the client from their preferred slots:

A. The whole idea behind the client categorization into slots (morning, noon, evening, late evening) was to find the measure of total deviation a client has throughout their digital PIPO logs. To achieve the same, I initiated two variables namely **Jump** and **workout_count**. Both the initiated variables will have an initial value of 0 and the value will be incremented based on some logical criteria, In the end, both the variables will end up having the following information:

jump: The count of the times a client jumped (in another slot) from his/hers preferred slot. For example, let's assume for a morning slot client XYZ (XYZ's preferred slot is the morning slot) if the jump value comes out to be 30, then it will mean that the client deviated from the morning slot 30 times throughout the time he/she visited the gym, thus jump variable will have the jumps each and every client made through his history.

workout_count: the count of the times a client entered the gym for his workouts throughout their history, for example let's assume there is a client who bought a 1-month gym subscription on the 1st of January and then never recharged his subscription, the client shows up 23 times in the entire month of January out of the 31days he paid for, these 23 times are recorded in the PIPO dataset, and 23 corresponds to none other than the variable **workout_count**

After having the values corresponding to both these variables for each and every client, a new variable called **deviation** is initialized the value this variable holds will be calculated as follows:

deviation = (jump / workout_count) * 100

The deviation indicates the measure of flexibility (in %) a client has. The deviation will be calculated in an iterative fashion for each and every unique client, every corresponding deviation will be appended as a value to a key, and the key will again correspond to the unique #Rno thus giving birth to a new dictionary which will be labeled as **client_deviation_dict** which will be of the format:

- ❖ **client_deviation_dict = {#Rno: int type value, #Rno: int type value, ….}**

where the integer type value is nothing but a value corresponding to the **deviation variable.**

Hence now we have ended up with a dictionary that gives us the exact information about the deviation of each and every client.

B.

- The above thought process will be actualized by tracing the list **client_unique** inside of which we again trace **the df_pipo,** every Registration number of the unique client list will be matched against the Registration number inside of the df_pipo DataFrame, If the match is found the value of the variable **workout_count is incremented by 1**.
- As soon as the value of the workout_count variable is incremented, the preferred slot of the matched client is accessed using the **client_preferred_slot_dict,** now we know what slot the current client in the current iteration belongs to, this information is saved in another variable called **unique_client_slot**
- After knowing the slot, a client belongs to we can verify with an IF condition whether or not the current client in the current iteration of the df_pipo belongs to his preferred slot or not, **if** the value of **I'th index** of the **hour column in the df_pipo** does not equate to the value in the variable **unique_client_slot** then the **jump variable is incremented by 1** Else we continue the df_pipo loop and look at the next entry or next digital log of the matched client and repeat the process.
- At the end of the **first iteration** of the outer loop (outer loop is the **For loop** which traces every unique client), we will have the **deviation of the first unique client**, the deviation is then allotted to **client_deviation_dict**
- This whole process is repeated 494 times as 494 elements in the list are traced by the outer for loop and the deviation for each client will be saved in a new dictionary called **client_deviation_dict.**

STEP 3. Identification of Flexible clients

A. We now possess the measure of the deviation of 494 unique clients in form of a dictionary. The next mountain to climb is to sort the clients into two categories, flexible and non-flexible. Categories will be determined based on the **deviation tolerance level**, clients above the deviation tolerance level will be labeled as flexible clients

B. To achieve the current goal, we need to now trace the **client_deviation_dict**, every key and the corresponding value of that key will be traced via For loop, and if the corresponding value of the key (value: the measure of deviation in %) is greater than the set tolerance of 25% then the client will be appended to a newly created list called

total_flexible_clients_list, this simple logic will be executed iteratively for 494 times and all the clients who deviated more than 25 percent of their journey with the track fitness club they worked out will be added to the **total_flexible_clients_list** and thus flexible client identification is carried out.

C. It is also important to notice that the sole purpose of client categorization, and flexible client identification is to entice some clients with special personalized offers where a client may pay a discounted amount for the gym subscription which enables them to visit the gym only during the noon slot, to achieve the ultimate goal a mere list of flexible clients isn't enough for following reasons:
    - A flexible client might not be always willing to restrict themselves to any specific slot for their workout, for example, let's assume a client prefers the morning slot but the client also deviates more than 40 percent of the time from the preferred morning slot, which is supposed to be a good thing from the analytical viewpoint, right? , No! and here's why the client deviated 40 percent of the time but what matters the most is to which slot deviating clients deviate to, in the case above it was later discovered that this particular client deviated to the late evening slot, the late evening slot consists of prime hours like 6 pm and 7 pm, hence a flexible client **who prefers morning slot** deviates to late evening slots and vice a versa is also true a client who prefers late evening slot deviates to the morning slot, no matter where these early birds and late owls deviate to they always create overcrowding
    - The flexible clients who preferred morning and late evening slots are mostly working professionals who are currently working from home or university students although technically they may appear to be flexible but the flexibility, they offer is mostly useless from a practical viewpoint it is difficult to sway such clients with incentives
    - However, the opposite can be said about clients who prefer the noon or evening slot, a client who is flexible and prefers a noon slot (11 am, 12 pm, 1 pm,2 pm, 3 pm) or evening slot (4 pm, 5 pm) is most likely to be incentivized which will enable them not to deviate to peak hours and stay in their preferred slots.

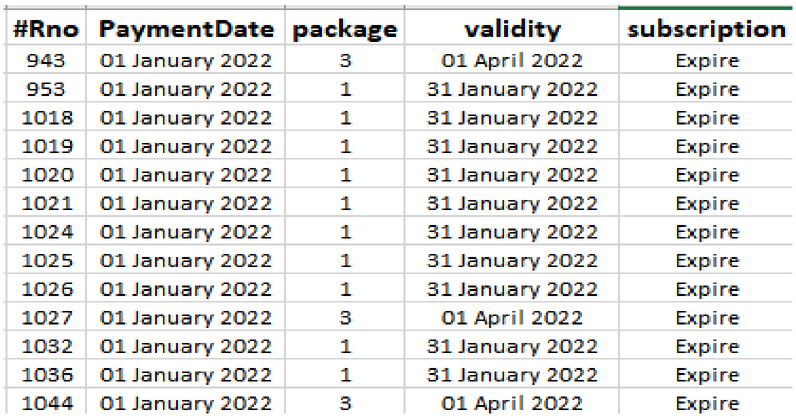
D. and so, the **updated goal** now is to generate a new list of clients where:
    - ✓ The client is flexible
    - ✓ The client prefers the noon or evening slot
    - ✓ The client is currently active in the gym

The clients with the above 3 features are the **special flexible clients**. To achieve the new goal, we need to sort out these clients from the **total_flexible_clients_list**.

STEP 4. Identification of special flexible clients

A. The identification of flexible clients is already completed, the second priority is to identify clients who prefer noon and evening slot, this is done via **client_preferred_slot_dict,** the dictionary contains the key-value pairs, a key corresponding to the unique #Rno and value to a corresponding preferred slot, the dictionary is traced via for loop when a client with a noon or evening slot shows up, the #Rno of the client(key) is appended in a **new list** called **noon_eve_client_list.** Thus, the sorting of clients with the required feature is achieved

B. The final feature of the flexible client is that the client must be currently active, hence a list of active clients is generated by tracing the df_invoice (the invoice datasheet)

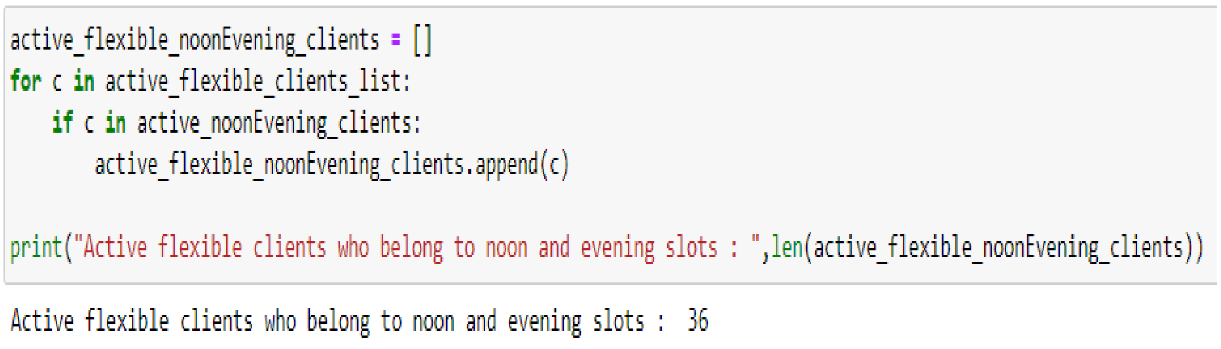


#Rno	PaymentDate	package	validity	subscription
943	01 January 2022	3	01 April 2022	Expire
953	01 January 2022	1	31 January 2022	Expire
1018	01 January 2022	1	31 January 2022	Expire
1019	01 January 2022	1	31 January 2022	Expire
1020	01 January 2022	1	31 January 2022	Expire
1021	01 January 2022	1	31 January 2022	Expire
1024	01 January 2022	1	31 January 2022	Expire
1025	01 January 2022	1	31 January 2022	Expire
1026	01 January 2022	1	31 January 2022	Expire
1027	01 January 2022	3	01 April 2022	Expire
1032	01 January 2022	1	31 January 2022	Expire
1036	01 January 2022	1	31 January 2022	Expire
1044	01 January 2022	3	01 April 2022	Expire

Fig: Invoice datasheet

A new list consisting of unique #Rno of all the active clients will be constructed by iteratively tracing the DataFrame above, checking the value in the subscription column, and appending the Registration number from the #Rno column to the list if the value in the subscription columns corresponds to "Active". The new list is labeled as **active_client_list**

C. The intersection of the clients from the **active_client_list, noon_eve_client_list,** and **total_flexible_clients_list** are the special flexible clients, which is done via the following logic:



```
active_flexible_noonEvening_clients = []
for c in active_flexible_clients_list:
    if c in active_noonEvening_clients:
        active_flexible_noonEvening_clients.append(c)

print("Active flexible clients who belong to noon and evening slots : ",len(active_flexible_noonEvening_clients))
```

```
Active flexible clients who belong to noon and evening slots :  36
```

Hence the special flexible clients are isolated, and the list of these clients is been sent to the responsible staff of the fitness studio. **There are 36 such clients**

2. <u>Membership Retention</u>:

The fitness studio struggles with Membership Retention, to understand why the business fails to retain clients so much so that the entire revenue which is generated every month fluctuates in an order of tens of thousands of rupees where the difference between revenues of January and march is a whooping sum of 1 lac 30 thousand, an appropriate measure called as **Adherence Rate** (**AR**) is quantized.

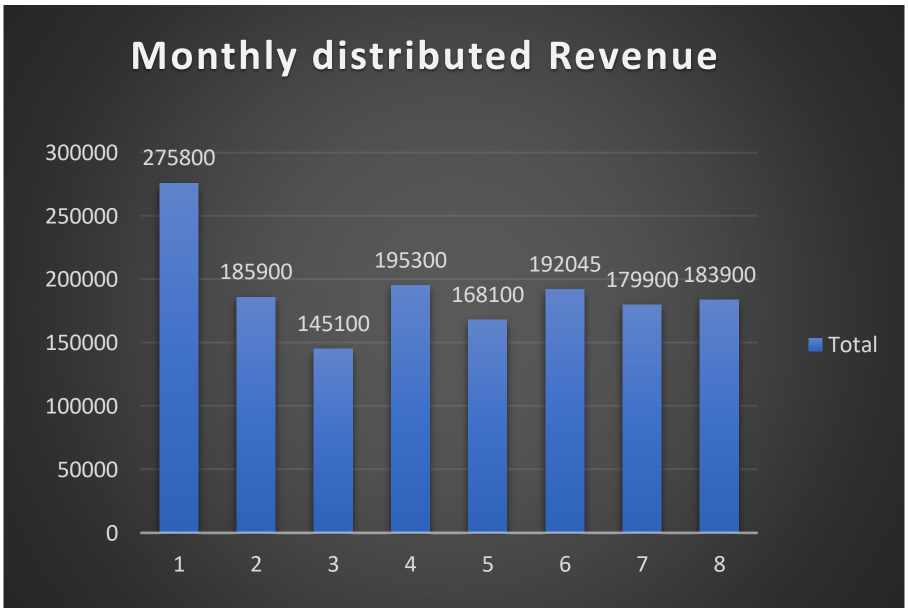
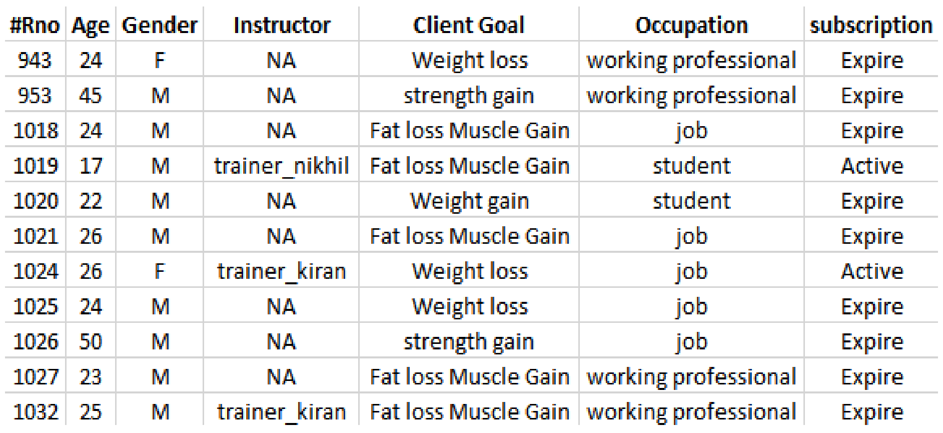


Fig: x-axis corresponds to the Month and the y-axis corresponds to the Rupee

Adherence rate is a medical term that is defined as the proportion of time times a prescribed dose is taken during a medical trial, in the fitness business the AR measures the **proportion of times a client visits the gym** for a workout, a client may buy a subscription for 3 months but the number of times he/she workouts within those 3 months is the adherence rate of the respective client.

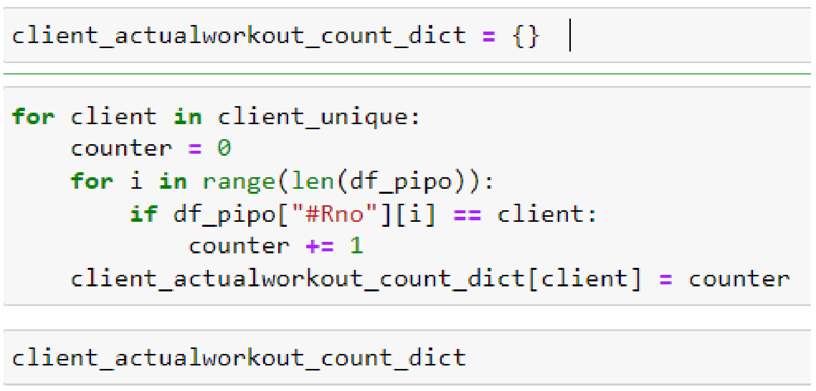
The adherence rate for every client is calculated and those clients who adhered to the gym more than 75% of the time are labeled as **High AR clients,** after the most consistent people in the history of the track fitness club are known, the other factors associated with these high AR clients can be observed from the **PII datasheet**. The PII datasheet is one of the data files I collected from the business, PII stands for Personally Identifiable Information an image of the data file is given for reference below.



#Rno	Age	Gender	Instructor	Client Goal	Occupation	subscription
943	24	F	NA	Weight loss	working professional	Expire
953	45	M	NA	strength gain	working professional	Expire
1018	24	M	NA	Fat loss Muscle Gain	job	Expire
1019	17	M	trainer_nikhil	Fat loss Muscle Gain	student	Active
1020	22	M	NA	Weight gain	student	Expire
1021	26	M	NA	Fat loss Muscle Gain	job	Expire
1024	26	F	trainer_kiran	Weight loss	job	Active
1025	24	M	NA	Weight loss	job	Expire
1026	50	M	NA	strength gain	job	Expire
1027	23	M	NA	Fat loss Muscle Gain	working professional	Expire
1032	25	M	trainer_kiran	Fat loss Muscle Gain	working professional	Expire

I followed a **4-step action plan** to find the leaks which currently restrict the business from retaining more and more of its clients

1. Create a dictionary consisting of key-value pairs where the key corresponds to #Rno and the value corresponds to the number of times a client visits the studio, this dictionary is labeled as **client_actualworkout_dict.**



```
client_actualworkout_count_dict = {}
```

```
for client in client_unique:
    counter = 0
    for i in range(len(df_pipo)):
        if df_pipo["#Rno"][i] == client:
            counter += 1
    client_actualworkout_count_dict[client] = counter
```

```
client_actualworkout_count_dict
```

The above block of codes traces every unique client from the **client_unique list** on the outer for loop, a counter variable is set to 0 every time a new unique client appears from the list, the inner for loop traces the df_pipo DataFrame, the value of the counter is incremented every time the unique client from the list and the registrations number (from the df_pipo) indicating the unique client **is matched**, at the end of each iteration the counter carries a number corresponding to the number of times a client visited the gym. the counter is set as the value of the **client_actualworkout_dict.**

2. Step 2 involves assembling another dictionary called **client_totalDuration_dict,** this dictionary has a key corresponding to #Rno and **values corresponding to the total duration of time (in days) the client bought the gym subscription**.

```python
client_totalDuration_dict = {}
```

```python
for c in client_unique:
    client_allRecharges_duration_list = []
    for i in range(len(df_invoice)):
        if df_invoice["#Rno"][i] == c:
            client_allRecharges_duration_list.append(df_invoice["package"][i])

    client_totalDuration_dict[c] = (sum(client_allRecharges_duration_list))*30  # in days
```

The df_invoice data set is the collection of all the invoices for a period of 8 months, every time a client renews his subscription for a package, the package value is appended to a list, and at the end of the iteration, the dictionary is appended with a value which is nothing but the sum of all the packages in the list which corresponds to the total months a client paid the price for, this iterative process is again carried out for all the 494 unique clients.

3. From the virtue of both the dictionaries in the above two steps, we can finally find the Adherence rate of each and every client.

```python
client_dedication = {}  # key : Rno and value : duration (in %) of time a clients workouts in the gym
```

```python
for key1,actual_duration in client_actualworkout_count_dict.items():
    for  key2,total_duration in client_totalDuration_dict.items():
        if key1 == key2:
            client_dedication[key1] = round((actual_duration/total_duration)*100,3)
```

A new dictionary labeled as **client_dedication** is initialized and the same iterative process is carried out to set the values of this newly created dictionary, both the earlier created dictionaries are traced out via for loop and as soon as the key (#Rno) matches, the matched key is set as the new key of the **client_dedication** dictionary and the value corresponding to this new key is the ratio of values of actual duration and total duration which is nothing but the AR.

And thus, we have a final dictionary that shows how dedicated a client is toward his fitness goals, every client who has an AR value of greater than 75 is labeled as a High AR client which is shown below.

4. Identification of High AR clients:

```python
high_AR_clients = [] # list of high Ar clients
for k,v in client_dedication.items():
    if v>75:
        high_AR_clients.append(k)

len(high_AR_clients)
```

```
35
```

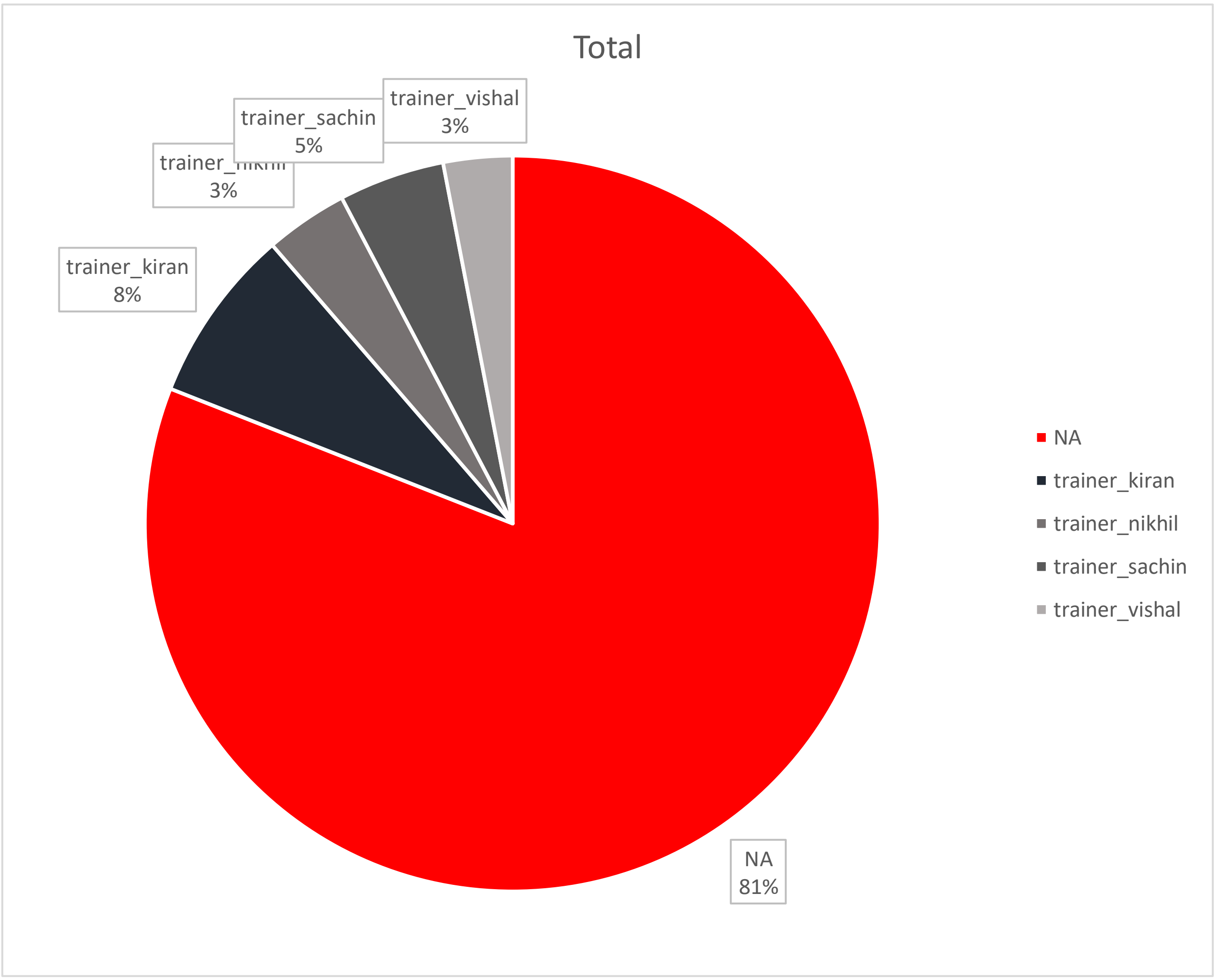
The client_dedication dict is traced and every value corresponding to Adherence Rate is compared with 75, if the rate is higher than 75 then the client is added to a list called **high_AR_clients, there are 35 such clients** these are the clients who are absolutely serious about their workouts, the other features of these clients like age, gender, Goal, Instructor can be now minutely observed to draw conclusions in the result and finding section of the report

Results And Findings:

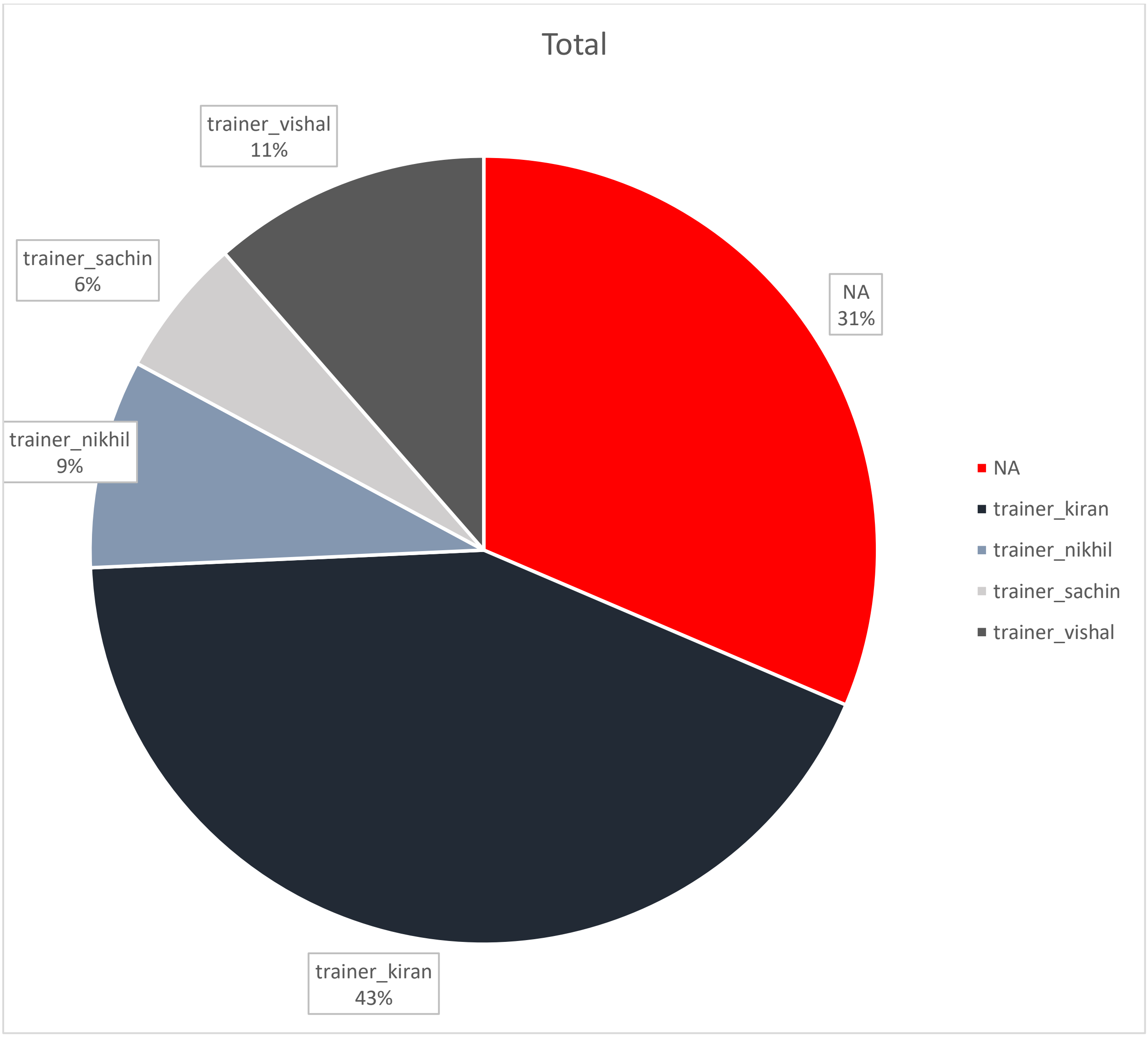
After all the heavy lifting in the analysis section, I am left with the most precious extracted pieces of data both in the **form of a list** namely

- ✓ **high_AR_clients**
- ✓ **total_flexible_clients_list**

I used both pieces of information to put two and two together and came up with the following charts



The above is a pie chart derived from the given PII data at a first glance no astonishing conclusion could be made from it, a tiny percentage of clients opt for personal training **with trainer Kiran grabbing the largest slice of the pie chart with 8%,** a total of 19% of the clients in the history of track fitness club opted for personal training which seems normal unless u take a look at the pie chart below.



The above pie chart is plotted only for High AR clients (whereas the previous pie chart corresponds to All the 494 clients throughout the history of the track fitness club), After the identification of **high AR clients** (in form of the list) I converted the list into a DataFrame and exported the DataFrame to excel, then I filtered out the high AR clients in the PII dataset, and copied them in a new sheet, in this new excel sheet I plotted the pie chart which can be seen above.

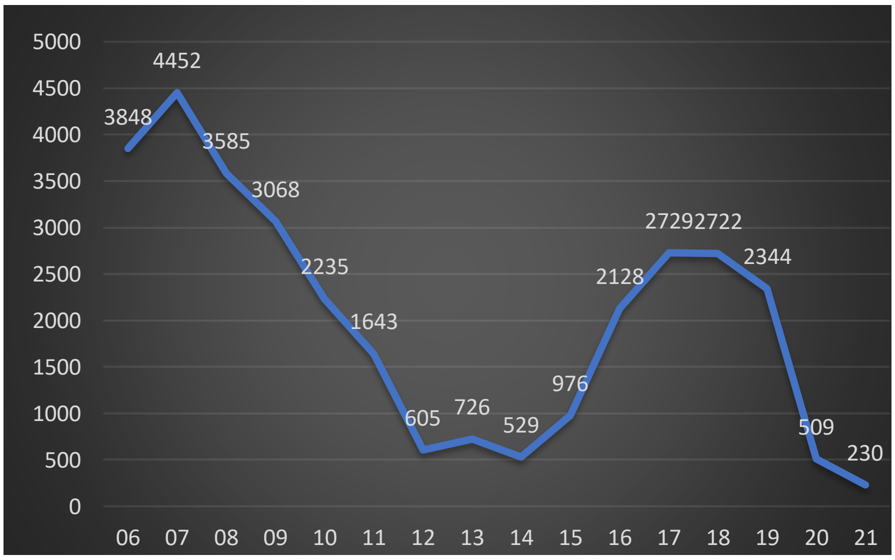
- A whopping 69% of total clients within the high AR clients have personal trainers

- Trainer Kiran has almost grabbed half the clients within the high AR list.
- Trainer Kiran has about **43%** of the total high AR clients, there are 35 such clients which brings about **15 clients for trainer Kiran,** so it is safe to say that**, trainers Kiran is highly correlated with clients with a High Adherence Rate**

- It was later found out that **77%** (27 clients out of 35 high AR clients) are currently active in the gym for a long period
- Though from a bird's eye point of view there was no theoretical correlation between having a personal trainer and being a high AR client who is also active, but with the help of retention analysis, now it can be said with certain confidence that if a client opts for personal training, especially at the beginning of their journey, then they are comparatively more likely to become a high AR client, a high AR is a client who is active since a long time, hence they are also the most retained clients.

- Hence if clients at the time of registration opt for basic personal training they will most likely stick with the gym for a long time and thus the client retention problem can be solved by making them invest as much as to the brim of the capacity.

Coming back to the overcrowding issue at hand, I took the liberty of simulating the PIPO dataset with one exception which is that all the flexible clients have been restricted to the noon slot, after plotting the Line chart of the newly simulated PIPO data sheet, it can be noticed that the frequency at peaks is significantly reduced



The peaks at 7 am and 5 pm have been significantly reduced, the above graph is the graph that can be obtained not only in the simulation but only in the real case if the recommendations are followed not as a one-time solution but as a routine. Overcrowding simply means that the business is not actively trying to do anything about it, a service business is all about the clients and not the product. Clients had to be made feel special and exclusive to retain them for longer periods in addition to this, it is of peak importance that the clients also get to see results for their set goals.

Interpretation of results and Recommendations:

The whole crux of the **Retention analysis** can be explained as follows:

- Personal training → Trainer Kiran ➔ High AR client → **77% of High AR clients are Active for more than 6 months → clients are retained**

- The gym needs to make efforts towards getting a personal trainer hired by the client at the time of client registration, when a new client enters the gym, they can be convinced to hire a personal trainer, **hence my recommendation to the business will be to run a subscription + personal training offer rather than investing in monsoon or summer offers which are a waste of capital.**

- **To avoid overcrowding at peak hours the business needs to send an SMS to all the special flexible clients that I have identified thus alerting them of the special personalized offers exclusively made available for them**

- **Solving The Overcrowding issue is a marathon and not a sprint, a one-time solution is not enough to deal with this problem, special flexible clients had to be identified and contacted every month for as long as any significant results are seen.**

- In the coming few months I will host a **web app** and share it with the responsible staff of the track fitness club, what the website will do with a single push of a button is to identify all the **active flexible noon evening slot preferring clients** (special flexible clients) so that they can be later contacted via their phone number through an automated SMS service and made aware of the personalized offer exclusively from them.