PORTFOLIO PROJECT

Submitted by –
Hashim Ali Zaidi

PROFESSIONAL BACKGROUND

I am Hashim Ali Zaidi, currently pursuing *Master of Computer Applications* from *Kamla Nehru Institute of Technology, Sultanpur.* I will be graduating in *May, 2023* and I have scored *8.12 CGPA* till now.

I'm a multidisciplinary professional with expertise in:

- Data analysis
- Python programming
- Machine learning
- Excel
- SQL
- Data science
- Web Development

I'm a quick learner and passionate about staying up-to-date with the latest trends and technologies. With my diverse skill set, I am capable of working on a wide range of projects from end-to-end.

Whether it's analysing complex data, building scalable web applications, or implementing machine learning algorithms, I'm up for the challenge. I have excellent communication and collaboration skills and enjoy working in a team environment to drive project success.

INDEX

S. No.	Name of Project	Page No.
	What is Data Analytics	4
1	Data Analysis Process	5-8
2	Instagram User Analytics	9-17
3	Operation and Metric Analytics	18-26
4	Hiring Process Analysis	27-32
5	IMDb Movie Analysis	33-43
6	Bank Loan Case Study	44-71
7	XYZ Ads Airing Report	72-81
8	ABC Call Volume Trend	82-90

What is Data Analysis

Data analysis is a critical process that involves inspecting, cleaning, transforming, and modelling data to discover meaningful information and support decision-making. It is a multidisciplinary field that combines various statistical and computational techniques to extract insights from data.
The importance of data analysis cannot be overstated. With the massive amount of data being generated by individuals, businesses, and governments on a daily basis, the need for skilled data analysts has never been greater. Organizations that can effectively collect, store, analyse, and interpret data can gain valuable insights into customer behaviour, market trends, and operational efficiency, giving them a competitive edge in their respective industries.

The data analysis process typically involves several steps. The first step is data collection, where data is gathered from various sources, such as databases, surveys, or sensors. Once the data is collected, it is cleaned and pre-processed to remove any errors or inconsistencies. The data is then organized and transformed to prepare it for analysis.
The next step is data exploration, where data analysts use various statistical techniques to identify patterns, trends, and correlations in the data. This may involve creating visualizations, such as charts or graphs, to help identify key insights.

In conclusion, data analysis is a critical process that plays a vital role in modern organizations. By leveraging data analysis techniques, organizations can gain valuable insights into customer behaviour, market trends, and operational efficiency, giving them a competitive edge in their respective industries. Data analysts play a crucial role in this process, using their specialized skills and expertise to transform raw data into meaningful insights that drive business decisions.

Module - I

Data Analytics Process

Description:

We use Data Analytics in everyday life without even knowing it. For e.g. : Going to a market to buy something.

- Plan - We first decide which things I need before going to market . Is it a shirt , jeans , footwear etc.

- Prepare - Next I need to check how much I am willing to spend and how to get that money.

- Process - Then I need to check how much I want from the data. Like if I am going to buy footwear what do I want - slippers / shoes / sandals etc.

- Analyse - You obviously won't buy things which are out of trend, Also you need to check does the jeans which you have and the colour of t-shirt you want to buy, will it make a good combination.

- Share - Now you communicate your idea to the shopkeeper to find the best suitable fit for you.

- Act - Then you finally buy it!

Task:

Your task is to give the example(s) of such a real-life situation where we use Data Analytics and link it with the data analytics process. You can prepare a PPT/PDF on a real-life scenario explaining it with the above process (Plan, Prepare, Process, Analyse, Share, Act) and submit it as part of this task.

Plan:
The manufacturing company wants to reduce unplanned downtime of its production line due to equipment failure. It decides to implement a predictive maintenance program.

Prepare:
The company collects data on the performance and maintenance history of its equipment, as well as environmental factors that may affect equipment performance (temperature, humidity, etc.). The data is cleaned, formatted, and integrated into a centralized data repository.

Process:
The company uses statistical techniques and machine learning algorithms to process the data and identify patterns and relationships that can be used to predict when equipment is likely to fail.

Analyse:
The company analyses the processed data to determine the factors that are most important in predicting equipment failure and builds predictive models based on those factors.

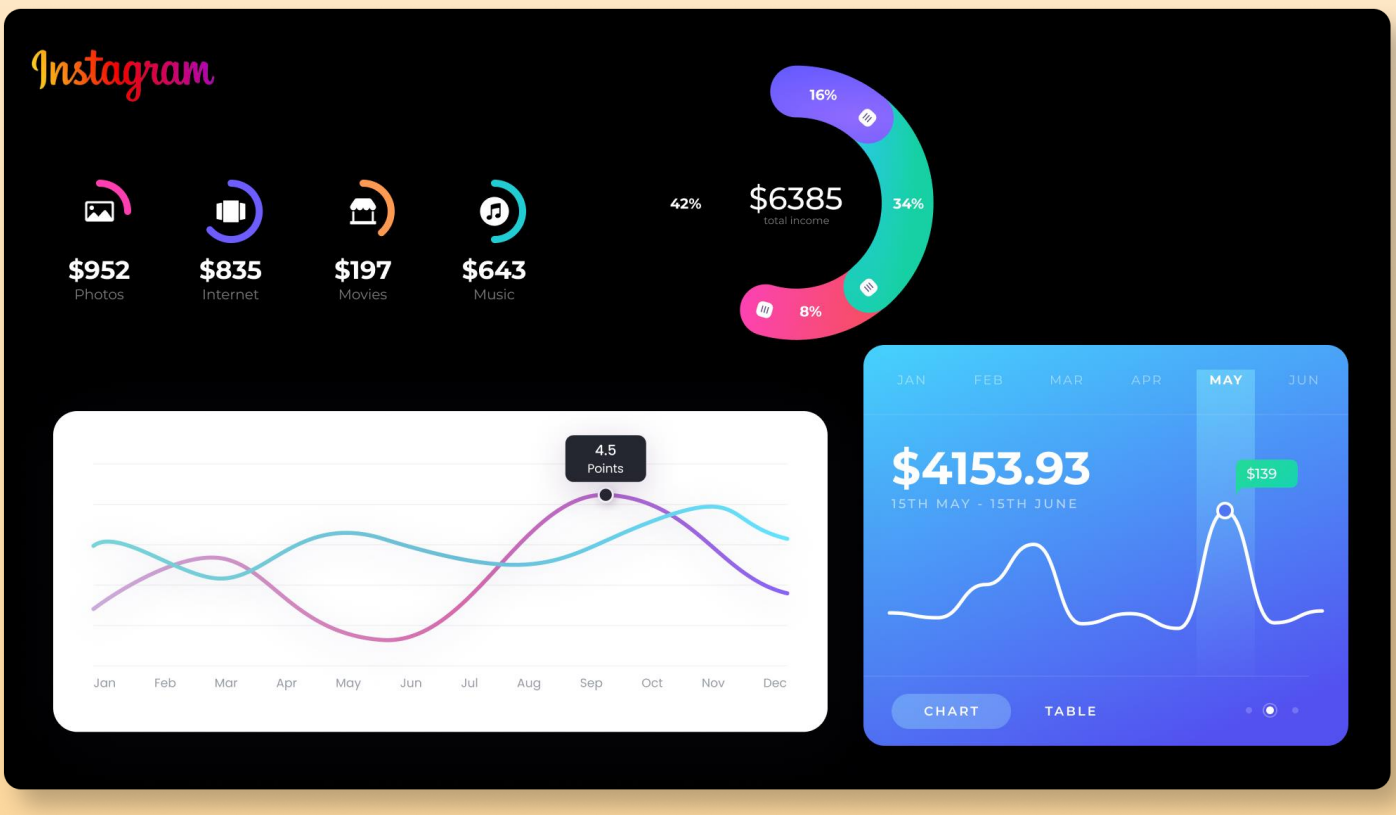
Share:
The company shares the results of the analysis with relevant stakeholders, including maintenance technicians and production managers. The predictive models are integrated into the company's maintenance management system.

Act:
The company uses the predictive models to schedule proactive maintenance activities, reducing the likelihood of unplanned downtime and improving equipment performance. The company continually monitors the results of the predictive maintenance program and adjusts the models as needed to ensure that they continue to provide accurate predictions.

This example shows how the data analytics process can be applied to real-life business problems and how it can help organizations make data-driven decisions.

Module - 2

INSTAGRAM USER ANALYTICS



Project Description :

The aim of this project is to perform analysis on Instagram user data and gain insights into the behaviour of Instagram users. SQL was used as a primary tool for data analysis and retrieval. The findings from the analysis provide valuable insights into user activity, preferences and demographics on the popular social media platform. This report presents the methodology and results of the analysis and discusses the implications of the findings for businesses and marketers."

Approach :

- Database Creation: Use SQL to create a database that can store the data and support the queries you need to perform the analysis.
- Data Loading: Load the data into the database.
- Data Analysis: Use SQL queries to perform the analysis and retrieve the insights you want to extract.

Tech-Stack Used :

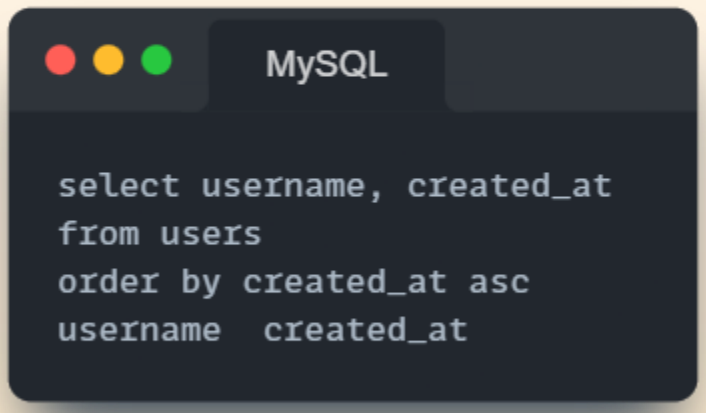
1. MySQL

Insights :

Below I have provided SQL queries and their respective outputs and with insights/outcomes for each of the 7 questions.

A) Marketing:

1 - Rewarding Most Loyal Users:



username	created at
Darby Herzog	06-05-2016 00:14
Emilio_Bernier52	06-05-2016 13:04
Elenor88	08-05-2016 01:30
Nicole71	09-05-2016 17:30
Jordyn.Jacobson2	14-05-2016 07:56

2 - Remind Inactive Users to Start Posting:

username	count(p.user_id)
Aniya_Hackett	0
Kasandra_Homenick	0
Jaclyn81	0
Rocio33	0
Maxwell.Halvorson	0
Tierra.Trantow	0
Pearl7	0
Ollie_Ledner37	0
Mckenna17	0
David.Osinski47	0
Morgan.Kassulke	0
Linnea59	0
Duane60	0
Julien_Schmidt	0
Mike.Auer39	0
Franco_Keebler64	0
Nia_Haag	0
Hulda.Macejkovic	0
Leslie67	0
Janelle.Nikolaus81	0
Darby Herzog	0
Esther.Zulauf61	0
Bartholome.Bernhard	0
Jessyca_West	0
Esmeralda.Mraz57	0
Bethany20	0

3 - Declaring Contest Winner:

MySQL

```mysql
select  u.username, u.id, count(l.photo_id)
from users as u
left join photos as p
on u.id = p.user_id
left join likes as l
on p.id = l.photo_id
group by u.username, l.photo_id, u.id
order by count(l.photo_id) desc
limit 1;
```

username	id	count(l.photo_id)
Zack_Kemmer93	52	48

Zack_Kemmer93 is the winner with most number of likes on a single photo.

4 - Hashtag Researching

MySQL

```mysql
select tag_name, count(t.id)
from photos as p
left join photo_tags as pt
on p.id = pt.photo_id
left join tags as t
on pt.tag_id = t.id
Group by t.id
order by count(id) desc
limit 5;
```

tag_name	count(t.id)
smile	59
beach	42
party	39
fun	38
concert	24

Here is a list of all the hashtags used along with the number of times they have been used.

5 - Launch AD Campaign

MySQL

```mysql
with cte as
( select
case
when weekday(created_at) = '0' then 'Monday'
when weekday(created_at) = '1' then 'Tueday'
when weekday(created_at) = '2' then 'Wedday'
when weekday(created_at) = '3' then 'Thursday'
when weekday(created_at) = '4' then 'Friday'
when weekday(created_at) = '5' then 'Saturday'
when weekday(created_at) = '6' then 'Sunday'
END as days_of_week
from users )
SELECT days_of_week, count(days_of_week)
from cte
group by days_of_week
```

days_of_week	count(days_of_week)
Thursday	16
Sunday	16
Tueday	14
Saturday	12
Wedday	13
Monday	14
Friday	15

Extracted a list of all the weekdays on which respective number of accounts were created.
“weekday()” function was used to extract weekday from DATETIME data type column.

B) Investor Metrics

6 - User Engagement:

```mysql
with cte as
( select u.id, count(p.id) as post_per_user
from users as u
left join photos as p
on u.id = p.user_id
group by u.id
order by post_per_user desc )
select post_per_user, count(post_per_user)
from cte
group by post_per_user
```

The above table show number of posts per user and how many users have posted each number of time. So there is only one user who has posted 12 times and two users who have posted 8 times and so on. From the extracted information we can say that an average user posts less than 6 times.

post_per_user	count(post_per_user)
12	1
11	1
10	1
9	1
8	2
6	1
5	14
4	13
3	9
2	13
1	18
0	26

```mysql
with cte as
( select u.id, count(p.id) as post_per_user
from users as u
left join photos as p
on u.id = p.user_id
group by u.id
order by post_per_user desc )
select avg(post_per_user)
from cte
```

The total number of photos on Instagram/total number of users = 2.57

avg(post_per_user)
2.57

7 - Bots & Fake Accounts:

MySQL

```mysql
with cte as
( select l.user_id, count(l.user_id) as number_of_likes
from photos as p
left join likes as l
on p.id = l.photo_id
left join users as u
on p.user_id = u.id
group by l.user_id )
select user_id, u.username,number_of_likes
from cte
left join users as u
on cte.user_id = u.id
where number_of_likes = 257
```

user_id	username	number_of_likes
5	Aniya_Hackett	257
14	Jaclyn81	257
21	Rocio33	257
24	Maxwell.Halvorson	257
36	Ollie_Ledner37	257
41	Mckenna17	257
54	Duane60	257
57	Julien_Schmidt	257
66	Mike.Auer39	257
71	Nia_Haag	257
75	Leslie67	257
76	Janelle.Nikolaus81	257
91	Bethany20	257

These are the accounts that have liked all the photos that are available, So we can conclude that these are bots.

Result :

Extracted many useful insights that could help in making data driven decisions for the business.

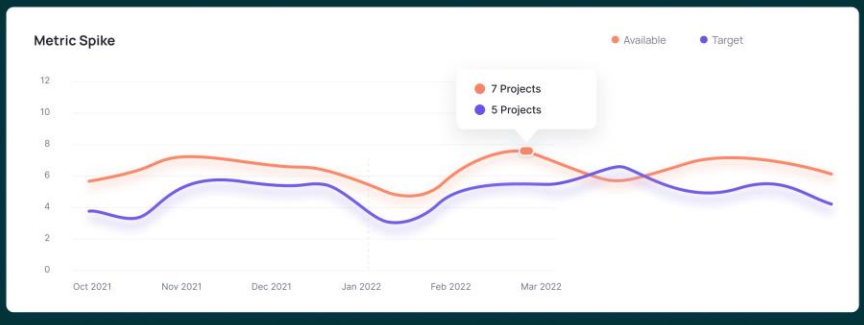
Used MySQL, had to study few advanced topics like CTEs and datetime functions.

Module - 3

Operational and Metrics Analytics

trainity

Operation Analytics & Investagating metric spike case study



Project Description :

This project involved using SQL to analyse operational data and investigate metric spikes. The goal was to generate insights and answer complex questions related to the organization's operations. I used a range of SQL techniques to answer complex questions, such as subqueries, joins, and aggregations. For example, I used subqueries to compare the performance of different teams or regions, and joins to combine data from multiple sources. I also calculated metrics such as rolling averages and percentiles to gain a deeper understanding of the data.

Approach :

- Database Creation: Use SQL to create a database that can store the data and support the queries you need to perform the analysis.
- Data Loading: Load the data into the database.
- Data Analysis: Use SQL queries to perform the analysis and retrieve the insights you want to extract.

Tech-Stack Used :

1. MySQL

Insights :

Below I have provided SQL queries and their respective outputs and with insights/outcomes for each of the questions.

Case Study 1 (Job Data):

A - Number of jobs reviewed:

```mysql
SELECT DATE_FORMAT(ds, '%Y-%m-%d %H:00:00') AS hour,
       (sum(time_spent)/count(*)) as Time_spent_per_hour_per_job
FROM jobs
WHERE MONTH(ds) = 11 AND YEAR(ds) = 2020
GROUP BY DATE_FORMAT(ds, '%Y-%m-%d %H:00:00');
```

hour	Time_spent_per_hour_per_job
30-11-2020 00:00	20
29-11-2020 00:00	20
28-11-2020 00:00	16.5
27-11-2020 00:00	104
26-11-2020 00:00	56
25-11-2020 00:00	45

B - Throughput:

```mysql
SELECT SUM(time_spent) / TIMESTAMPDIFF(SECOND, MIN(ds), MAX(ds)) AS throughput
FROM jobs;
```

throughput
0.0007

We cannot find 7 day moving average because the data provided contains only 5 days of data.

C - Percentage share of each language:

```mysql
select language, (count(*)/(select count(*) from jobs))*100 as perc
from jobs
group by language;
```

language	perc
English	12.5
Arabic	12.5
Persian	37.5
Hindi	12.5
French	12.5
Italian	12.5

Persian is used most number of times as compared to other languages

D - Duplicate rows:

We can see that there are no duplicate rows present in the table although if we check for every column separately we can find that some values are repeating. To find those values we can use the following syntax : -

```mysql
select ds
from jobs
GROUP BY  ds
HAVING COUNT(*) > 1;
```

DS
30-11-2020 00:00
28-11-2020 00:00

Case Study 2 (Investigating metric spike):

A - User Engagement:

MySQL

```mysql
SELECT
(SELECT COUNT(DISTINCT(user_id)) AS active_users
FROM users
WHERE state = 'active') / (SELECT COUNT(DISTINCT(user_id)) AS active_users
FROM users) * 100 as user_engagement_rate
FROM users
```

By running the above query we get to know that the user engagement rate is just at **49.2%.**

B - User Growth:

```mysql
SELECT
    new_users.device,
    avg(new_users.new_users / first_month.total_users*100) AS growth_rate
FROM (
    SELECT
        device,
        month(occurred_at) AS signup_month,
        COUNT(DISTINCT user_id) AS new_users
    FROM events
    GROUP BY device, signup_month
) AS new_users
JOIN (
    SELECT
        device,
        COUNT(DISTINCT user_id) AS total_users
    FROM events
    GROUP BY device
) AS first_month
ON new_users.device = first_month.device
group by new_users.device;
```

device	growth_rate
acer aspire desktop	36.49
acer aspire notebook	35.58
amazon fire phone	32.2
asus chromebook	34.89
dell inspiron desktop	35.56
dell inspiron notebook	36.38
hp pavilion desktop	35.3
htc one	33.37
ipad air	33.14
ipad mini	32.34
iphone 4s	34.7
iphone 5	35.11
iphone 5s	33.99

device	growth_rate
kindle fire	32.64
lenovo thinkpad	36.15
mac mini	39.41
macbook air	35.63
macbook pro	35.86
nexus 10	33
nexus 5	34.28
nexus 7	32.81
nokia lumia 635	36.25
samsumg galaxy tablet	31.85
samsung galaxy note	33.67
samsung galaxy s4	33.64
windows surface	32.35

Here’s a list of all the devices and their growth rate with respect to the first month.

C - Weekly Retention:

```mysql
SELECT
     WEEK(occurred_at) AS week,
COUNT(DISTINCT user_id) AS signups,
COUNT(DISTINCT CASE WHEN event_type = 'engagement' THEN user_id END) AS engagements,
COUNT(DISTINCT CASE WHEN event_type = 'engagement' THEN user_id END) / COUNT(DISTINCT user_id) *100 AS retention_rate
FROM events
GROUP BY WEEK(occurred_at)
ORDER BY week;
```

week	signups	engagements	retention_rate
17	740	663	89.5946
18	1260	1068	84.7619
19	1287	1113	86.4802
20	1351	1154	85.4182
21	1299	1121	86.2972
22	1381	1186	85.8798
23	1446	1232	85.2006
24	1471	1275	86.6757
25	1459	1264	86.6347
26	1509	1302	86.2823
27	1573	1372	87.2219
28	1577	1365	86.5568
29	1607	1376	85.6254
30	1706	1467	85.9906
31	1514	1299	85.7992
32	1454	1225	84.2503
33	1438	1225	85.1878
34	1443	1204	83.4373
35	118	104	88.1356

D - Weekly Engagement:

MySQL

```mysql
SELECT
      WEEK(occurred_at) AS week,
      COUNT(DISTINCT CASE WHEN event_type = 'engagement' THEN user_id
      END) / COUNT(DISTINCT user_id)/(select count(distinct(device))
      from events) * 100 AS engagement_rate_per_device
FROM events
GROUP BY WEEK(occurred_at)
ORDER BY week;
```

week	engagement_rate_per_device
17	3.44594594
18	3.26007326
19	3.32616102
20	3.28531572
21	3.31912122
22	3.30306912
23	3.27694435
24	3.33368195
25	3.33210312
26	3.31855023
27	3.35468727
28	3.3291059
29	3.29328419
30	3.30733159
31	3.29996951
32	3.24039784
33	3.27645233
34	3.20912628
35	3.38983051

E - Email Engagement:

```mysql
SELECT
  engagements,
  total_users,
  engagements/total_users*100 AS engagement_rate
from (  SELECT
        count(distinct(user_id)) AS total_users,
        COUNT(DISTINCT
            CASE
              WHEN action = 'email_open' THEN user_id
              WHEN action = 'email_clickthrough' THEN user_id END)  AS engagements
        FROM email_events) as counts
```

engagements	total_users	engagement_rate
5927	6179	95.9217

Result :

1. Extracted many useful insights that could help in making data driven decisions for the business.
2. Used MySQL, had to study few advanced topics like Window functions,CTEs and datetime functions.

Module - 4

Hiring Process Analytics

Project Description:

The project aims to analyse the hiring process of a company and provide insights. The analysis will be done using Excel and statistical techniques such as regression analysis and hypothesis testing.

Approach :

First of all I have removed the missing data (there was only one row containing missing data).

Secondly, I performed outlier analysis where I found there were 3 rows with very high salary and 2 rows with very low salary, so I deleted these 5 rows.

Then started analysing and answering questions.

Tech Stack used:

1. MS Excel

INSIGHTS :

A - Hiring:

To filter rows on the basis of gender that are hired, we can use COUNTIFS function in MS Excel.

Male : = COUNTIFS(D2:D7165,"Male",C2:C7165,"Hired")

Result : 2560

Female : = COUNTIFS(D2:D7165,"Female",C2:C7165,"Hired")

Result : 1856

B - Average Salary :

To find the average salary we can use AVERAGE function in MS Excel :

Average Salary : =AVERAGE(G2:G7163)

Result : 49892.13

C - Class Intervals :

We can directly make a class interval table using the data analysis tools present in MS Excel. To acces this tool -

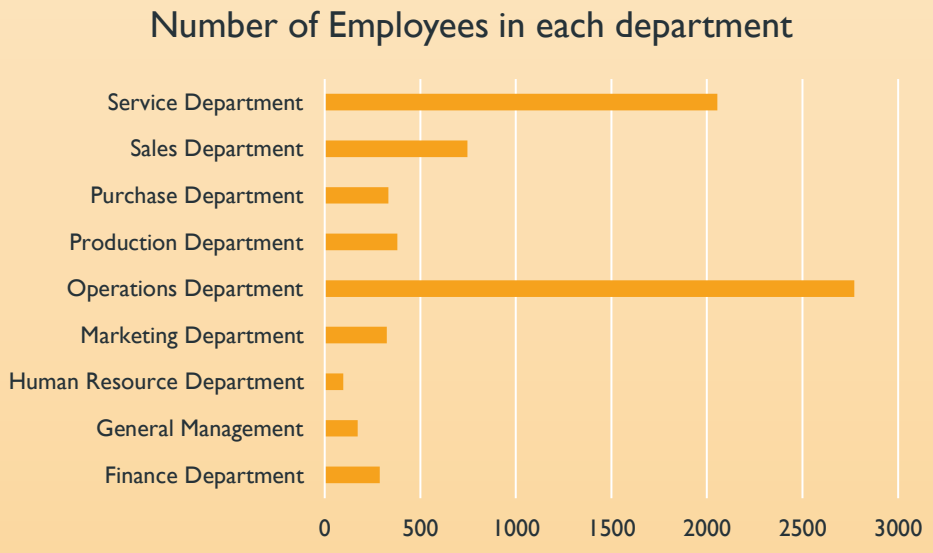
Data > Analyze > Data Analysis > Histogram

Bin	Frequency
10000	676
20000	732
30000	711
40000	710
50000	781
60000	750
70000	698
80000	734
90000	711
100000	659

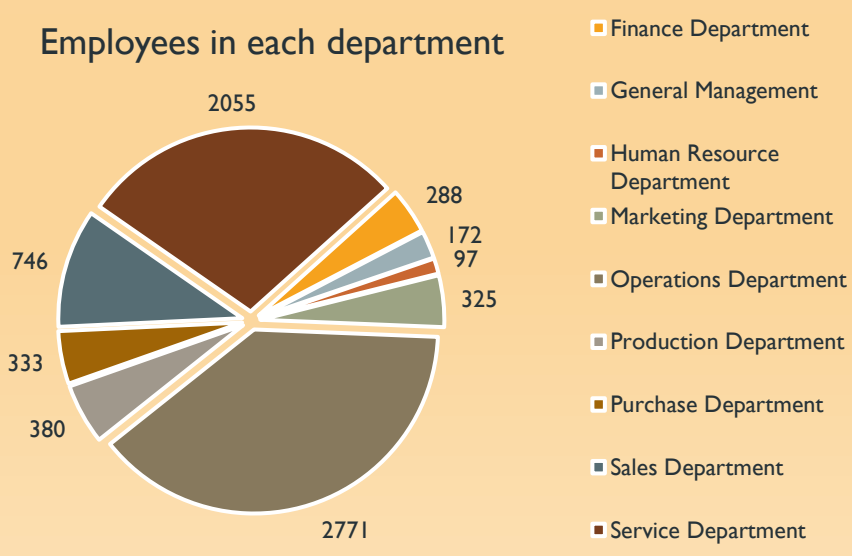
D - Charts and Plots:

We can make the required chart using pivot charts.
But first we have to make Pivot table that shows the *count* of employees in each department –
Now we can insert bar chart and pie charts.

Department	Count of Department
Finance Department	288
General Management	172
Human Resource Department	97
Marketing Department	325
Operations Department	2771
Production Department	380
Purchase Department	333
Sales Department	746
Service Department	2055



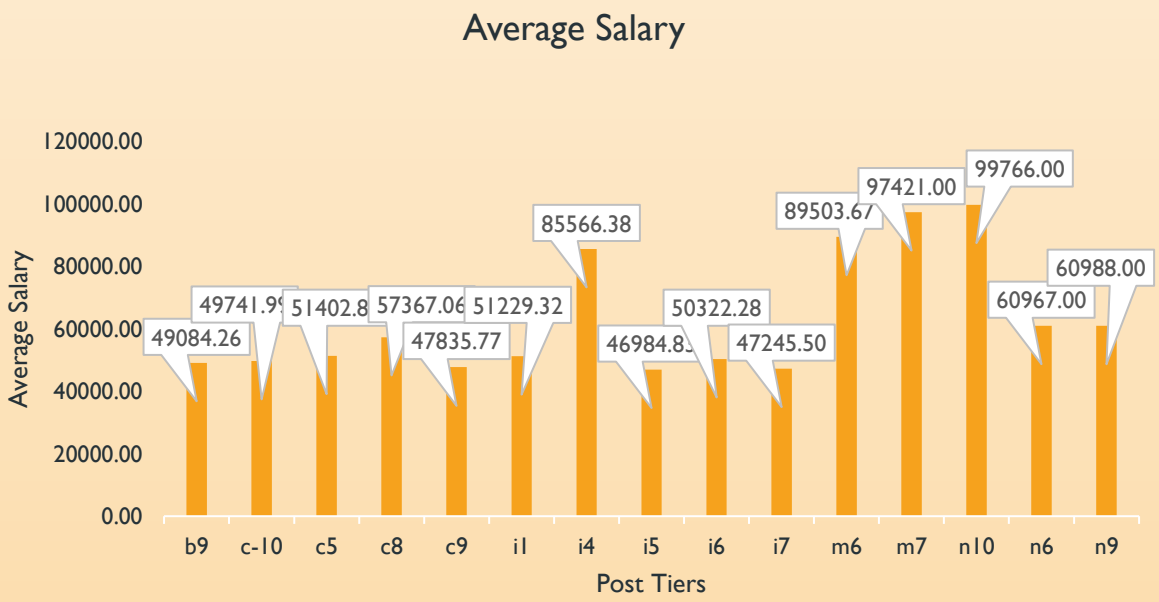
Bar Chart



Pie chart

E - Charts:

I have prepared a chart that shows average salary offered to each post tier.



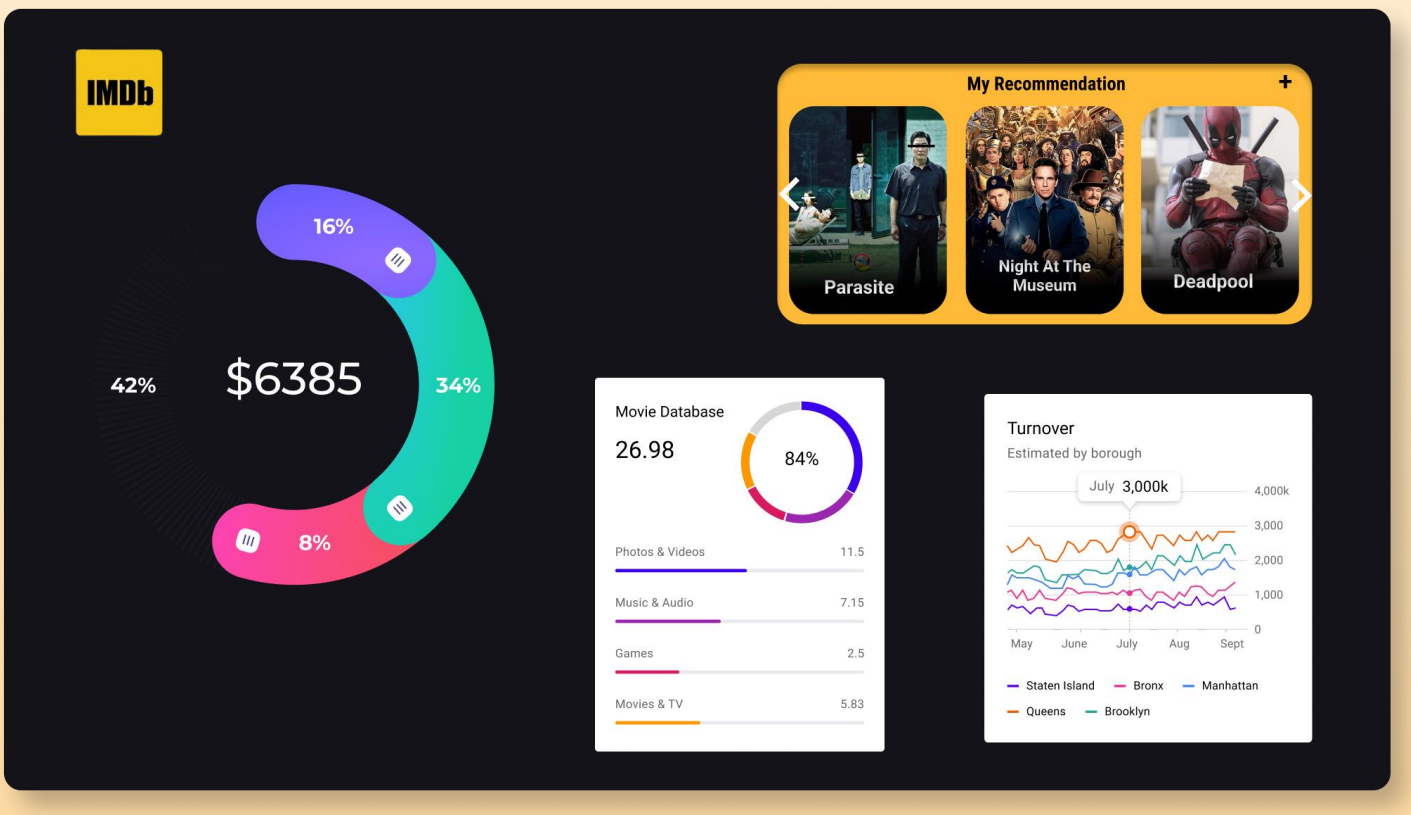
Results –

I have learned how to manipulate data in MS Excel, along with functions, pivot tables and charts & graphs.

This will help me for analyzing and performing EDA in MS Excel.

Module - 5

IMDb Movie Analysis



Project Description:

The project aims to analyse the hiring process of a company and provide insights. The analysis will be done using Excel and statistical techniques such as regression analysis and hypothesis testing.

Approach :

First of all I have removed the missing data (there was only one row containing missing data).
Secondly, I performed outlier analysis where I found there were 3 rows with very high salary and 2 rows with very low salary, so I deleted these 5 rows.
Then started analysing and answering questions.

Tech Stack used:

1. MS Excel

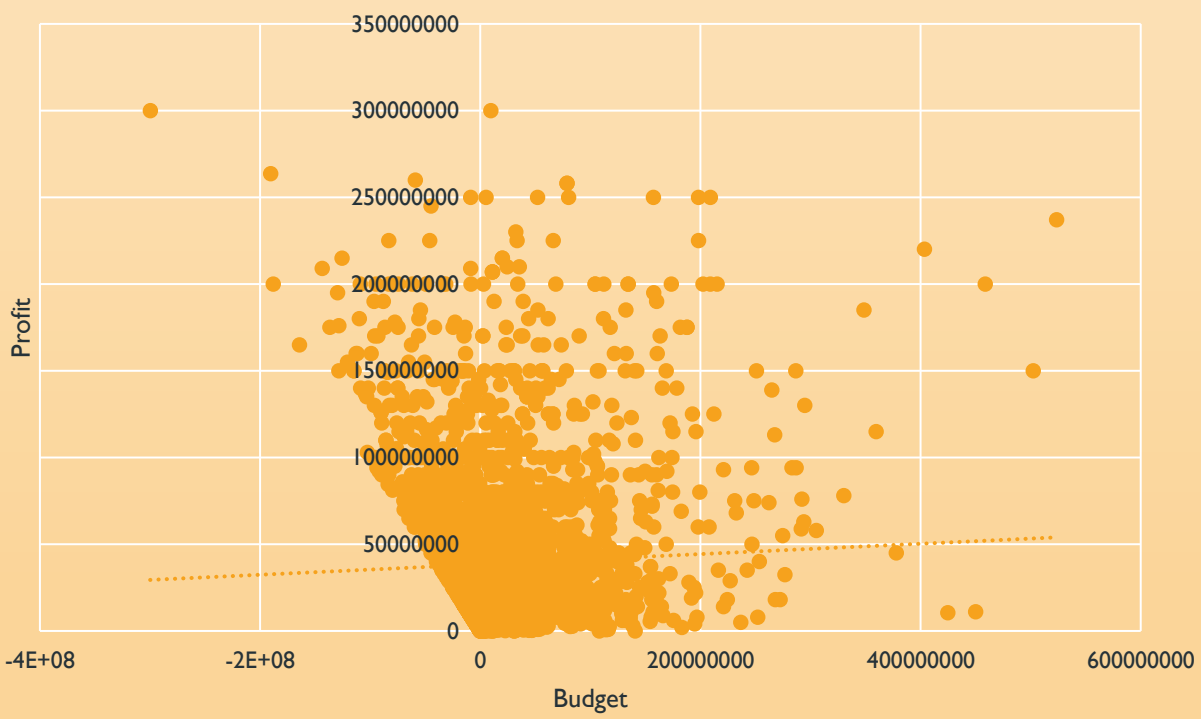
INSIGHTS :

A – Data Cleaning

- Dropped duplicate rows
- Deleted rows that contained blank cells
- Manipulated strings of movie titles and actor names
- Dropped columns that were useless
- Rechecked datatype of all the columns
- Removed outliers on the basis of budget and profit columns

B - Movies with highest profit:

To filter rows on the basis of gender that are hired, we can use COUNTIFS function in MS Excel.



The above chart shows the relationship between budget and profit. It has a very low positive correlation.

Highest Profit Movie : - **Avatar**

C - Top 250:

Here is a list of top 250 IMDB movies -

The Shawshank Redemption
The Godfather
The Dark Knight
The Godfather: Part II
The Lord of the Rings: The Return of the King
Pulp Fiction
Schindler's List
The Good, the Bad and the Ugly
Forrest Gump
Star Wars: Episode V - The Empire Strikes Back
The Lord of the Rings: The Fellowship of the Ring
Inception
Fight Club
Star Wars: Episode IV - A New Hope
The Lord of the Rings: The Two Towers
The Matrix
One Flew Over the Cuckoo's Nest
Goodfellas
City of God
Seven Samurai
Saving Private Ryan
The Silence of the Lambs
Se7en
Interstellar
The Usual Suspects
American History X
Modern Times
Spirited Away
The Lion King
Raiders of the Lost Ark
The Dark Knight Rises
Back to the Future
Terminator 2: Judgment Day
Gladiator
The Green Mile
Alien
Django Unchained
Apocalypse Now
The Departed
Psycho
Memento
The Prestige
Whiplash

The Lives of Others
Children of Heaven
Samsara
The Pianist
Star Wars: Episode VI - Return of the Jedi
American Beauty
Aliens
WALL-E
A Separation
Braveheart
Reservoir Dogs
Oldboy
Requiem for a Dream
Das Boot
Lawrence of Arabia
Once Upon a Time in America
Amlie
Princess Mononoke
Toy Story 3
Inside Out
Toy Story
The Sting
Indiana Jones and the Last Crusade
Good Will Hunting
Up
Unforgiven
Batman Begins
Inglourious Basterds
2001: A Space Odyssey
Amadeus
L.A. Confidential
Snatch
Some Like It Hot
Scarface
Eternal Sunshine of the Spotless Mind
Hoop Dreams
Room
Monty Python and the Holy Grail
The Hunt
Metropolis
Downfall
Raging Bull
Finding Nemo
Gone with the Wind
Captain America: Civil War
Gran Torino
A Beautiful Mind

Die Hard
How to Train Your Dragon
The Bridge on the River Kwai
Pan's Labyrinth
The Secret in Their Eyes
The Wolf of Wall Street
V for Vendetta
Trainspotting
On the Waterfront
Into the Wild
Lock, Stock and Two Smoking Barrels
The Big Lebowski
Incendies
The Act of Killing
Blade Runner
The Thing
Casino
Warrior
Howl's Moving Castle
The Avengers
Deadpool
Jurassic Park
The Sixth Sense
Monsters, Inc.
Pirates of the Caribbean: The Curse of the Black Pearl
Guardians of the Galaxy
The Help
Platoon
The Martian
The Bourne Ultimatum
Rocky
Gone Girl
Butch Cassidy and the Sundance Kid
The Imitation Game
Million Dollar Baby
The Truman Show
Groundhog Day
No Country for Old Men
The Revenant
Shutter Island
Stand by Me
Kill Bill: Vol. 1
12 Years a Slave
Annie Hall
Sin City
The Grand Budapest Hotel

The Terminator
Spotlight
The Best Years of Our Lives
The Wizard of Oz
There Will Be Blood
Prisoners
The Princess Bride
Woodstock
Hotel Rwanda
Mad Max: Fury Road
Amores Perros
Before Sunrise
The Celebration
In the Shadow of the Moon
Donnie Darko
Elite Squad
The Sea Inside
Rush
Tae Guk Gi: The Brotherhood of War
Akira
Jaws
The Exorcist
Aladdin
The Incredibles
Dances with Wolves
The Sound of Music
Rain Man
Slumdog Millionaire
The King's Speech
Catch Me If You Can
Star Trek
The Pursuit of Happyness
Doctor Zhivago
Black Swan
District 9
Young Frankenstein
Dead Poets Society
Mystic River
Ratatouille
Fiddler on the Roof
Kill Bill: Vol. 2
X-Men: Days of Future Past
JFK
The Artist
Sling Blade
Dallas Buyers Club
Boyhood

Bowling for Columbine
Casino Royale
Casino Royale
Sicko
Shaun of the Dead
Life of Pi
The Perks of Being a Wallflower
A Fistful of Dollars
Before Sunset
Central Station
Her
Waltz with Bashir
True Romance
Persepolis
Big Fish
The Straight Story
Brazil
In Bruges
Mulholland Drive
My Name Is Khan
Dancer in the Dark
Magnolia
Serenity
Cinderella Man
Blood In, Blood Out
Blood Diamond
The Iron Giant
Avatar
E.T. the Extra-Terrestrial
Shrek
Iron Man
Toy Story 2
Straight Outta Compton
Taken
Crouching Tiger, Hidden Dragon
Walk the Line
The Fighter
The Bourne Identity
Big Hero 6
My Fair Lady
Captain Phillips
Little Miss Sunshine
The Untouchables
Crash
Halloween
Halloween

Edward Scissorhands
The Hobbit: The Desolation of Smaug
How to Train Your Dragon 2
The Blues Brothers
Nightcrawler
Do the Right Thing
The Wrestler
Hot Fuzz
The Remains of the Day
Boogie Nights
The Hateful Eight
Once
Glory
Glory
Before Midnight
4 Months, 3 Weeks and 2 Days
Moon
Nine Queens
The Chorus
The Second Mother
Letters from Iwo Jima
The Right Stuff
Amour
Ernest & Celestine
Ed Wood
The World's Fastest Indian
Almost Famous
The Notebook
Hero
The Insider
Children of Men
Edge of Tomorrow

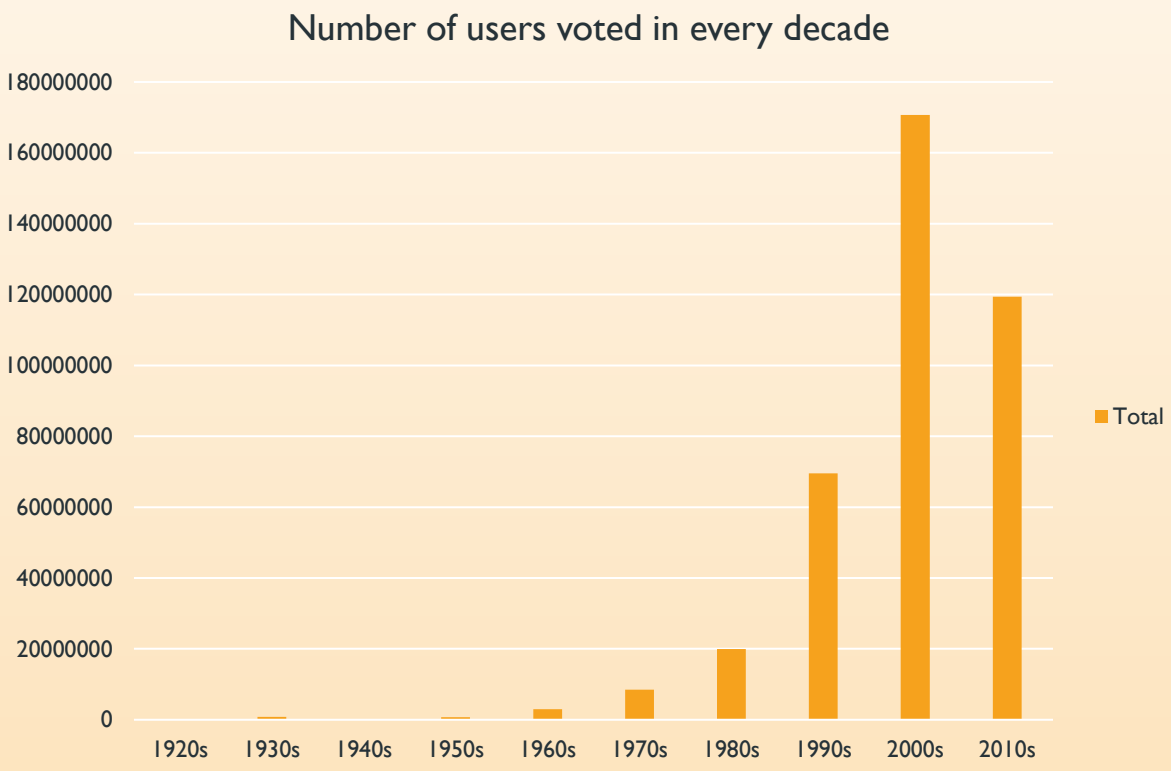
D - Best Directors :
List of Directors with highest average IMDB ratings -

Director Name	Average IMDB rating
Akira Kurosawa	8.7
Tony Kaye	8.6
Charles Chaplin	8.6
Ron Fricke	8.5
Majid Majidi	8.5
Damien Chazelle	8.5
Alfred Hitchcock	8.5
Sergio Leone	8.433333
Christopher Nolan	8.425
Asghar Farhadi	8.4

E - Popular Genres:
List of Genres with highest average IMDB ratings -

Genre	Aerage IMDB Rating
Crime|Drama|Fantasy|Mystery	8.5
Adventure|Animation|Drama|Family|Musical	8.5
Adventure|Animation|Fantasy	8.4
Adventure|Drama|Thriller|War	8.4
Documentary|Drama|Sport	8.3
Biography|Drama|History|Music	8.3
Adventure|Animation|Comedy|Drama|Family|Fantasy	8.3
Adventure|Drama|War	8.25
Drama|Mystery|War	8.2
Biography|Crime|Documentary|History	8.2

F - Charts:



The above chart shows the number of users that voted in each decade.

Best actor on the basis of highest average of num_critic_reviews :-

Actor Name	Average of num_critic_for_reviews
Albert Finney	750

Best actor on the basis of highest average of num_user_for_reviews :-

Actor Name	Average of num_user_for_reviews
Heather Donahue	3400

Results –

I have learned how to manipulate data in MS Excel, along with functions, pivot tables and charts & graphs.

This will help me for analyzing and performing EDA in MS Excel.

Module - 6

Bank Loan Case Study

Problem Statement

In this case study, we focus to perform EDA using visualizations and statistics summaries on loan application data. We need to find out what factors affect an applicant to be a defaulter or non-defaulter.

Approach

For EDA we will try to follow the bolow steps: 1) Import Modules

2) Read the dataset

3) Data Cleaning

On previous_application dataframe

- a) Missing value handling
  - i) identifying missing data
  - ii) Dropping missing data
- b) Outlier Analysis

On application_data dataframe

- a) Missing value handling
  - i) identifying missing data
  - ii) Dropping missing data

c) Data Imbalance

4) Analysis of different variables in segments :-

a) Segment 1

b) Segment 2

c) Segment 3

d) Segment 4

e) Segment 5

5) Correlation

a) Top 5 correlation for defaulters

b) Top 5 correlations for non-defaulters

6) Summary

1) Import Modules

Let's import libraries for EDA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Import our dataset as datframes using pandas library

2) Read Datasets

```
# reading application_data.csv
application_df = pd.read_csv('project 6 file 3.csv')
# reading previous_application.csv
prev_ap_df = pd.read_csv('project 6 file 1.csv')
```

```
print(application_df.shape)
print(prev_ap_df.shape)
```

```
(307511, 122)
(1670214, 37)
```

Lets check all features and it's datatypes

```
application_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YE
AR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

```
prev_ap_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   1670214 non-null  int64
 1   SK_ID_CURR                   1670214 non-null  int64
 2   NAME_CONTRACT_TYPE           1670214 non-null  object
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
```

```
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object
 16  NAME_CONTRACT_STATUS         1670214 non-null  object
 17  DAYS_DECISION                1670214 non-null  int64
 18  NAME_PAYMENT_TYPE            1670214 non-null  object
 19  CODE_REJECT_REASON           1670214 non-null  object
 20  NAME_TYPE_SUITE              849809 non-null   object
 21  NAME_CLIENT_TYPE             1670214 non-null  object
 22  NAME_GOODS_CATEGORY          1670214 non-null  object
 23  NAME_PORTFOLIO               1670214 non-null  object
 24  NAME_PRODUCT_TYPE            1670214 non-null  object
 25  CHANNEL_TYPE                 1670214 non-null  object
 26  SELLERPLACE_AREA             1670214 non-null  int64
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object
 30  PRODUCT_COMBINATION          1669868 non-null  object
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
```

Insights

```
# We will try to store lists of columns that are common in both dat
aframes and additional in any of them
additional_feat = []
common_feat = []

for col in application_df.columns:
    if col in prev_ap_df.columns:
        common_feat.append(col)
    else:
        additional_feat.append(col)

print(len(additional_feat))
print(len(common_feat))
print(common_feat)
```

```
114
8
['SK_ID_CURR', 'NAME_CONTRACT_TYPE', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOO
DS_PRICE', 'NAME_TYPE_SUITE', 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PRO
CESS_START']
```

Insights

3) Data Cleaning

Firstly I will try to perform data cleaning on the 'prev_ap_df'

a) Missing value handling

i) Identifying Missing Data

```
# define a function that has one arguement as dataframe
def missingdata_percentage(df):
    # create an empty dataframe with two columns
    missing = pd.DataFrame(columns=['category','percentage'])
    # iterate through all the columns of dataframe
    for col in df.columns:
        # a conditional statement that only passes those columns that have
at least 1 missing value
        if df[col].isna().values.any():
            # calulate the percentage of null values in that particular co
lumn
            percentage = 100*df[col].isna().sum()/df.shape[0]
            # append that column into to empty dataframe we created earlie
r
            missing = missing.append({'category' : col, 'percentage' : per
centage}, ignore_index=True)
    # return the dataframe we created
    return missing

missingdata_prev = missingdata_percentage(prev_ap_df)

missingdata_prev.sort_values('percentage', ascending=False)
```

	category	percentage
5	RATE_INTEREST_PRIMARY	99.643698
6	RATE_INTEREST_PRIVILEGED	99.643698
2	AMT_DOWN_PAYMENT	53.636480
4	RATE_DOWN_PAYMENT	53.636480
7	NAME_TYPE_SUITE	49.119754
10	DAYS_FIRST_DRAWING	40.298129
11	DAYS_FIRST_DUE	40.298129
12	DAYS_LAST_DUE_1ST_VERSION	40.298129
13	DAYS_LAST_DUE	40.298129
14	DAYS_TERMINATION	40.298129
15	NFLAG_INSURED_ON_APPROVAL	40.298129

```
3             AMT_GOODS_PRICE   23.081773
0                  AMT_ANNUITY   22.286665
8                  CNT_PAYMENT   22.286366
9          PRODUCT_COMBINATION    0.020716
1                   AMT_CREDIT    0.000060
```

Insights

There are 16 features in prev_app_df that have missing values. Permanently dropping the features (RATE_INTEREST_PRIMARY and RATE_INTEREST_PRIVILEGED) as 99% data is missing.

ii) Dropping missing data

```
prev_ap_df.drop(['RATE_INTEREST_PRIMARY','RATE_INTEREST_PRIVILEGED'], axis
=1, inplace=True)
prev_ap_df.dropna(subset=['AMT_CREDIT','PRODUCT_COMBINATION'], inplace=Tru
e)

#Checking the remaining columns
prev_ap_df
```

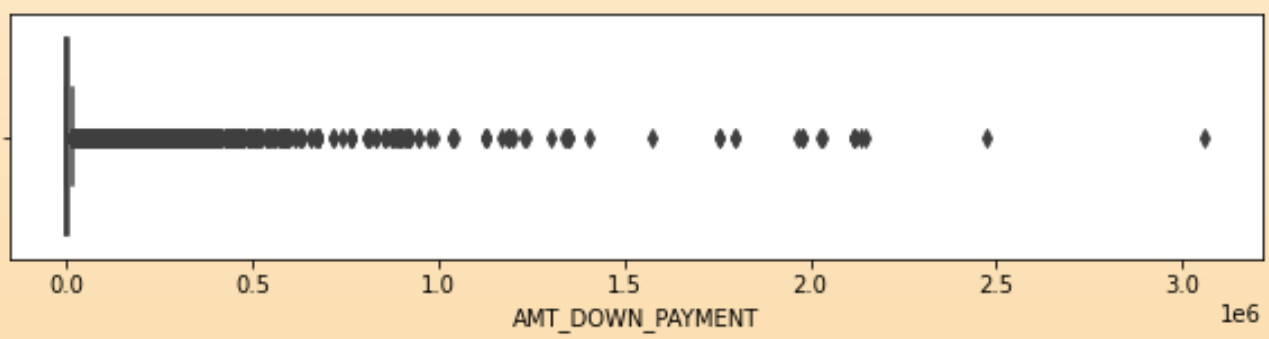
```
         SK_ID_PREV  SK_ID_CURR NAME_CONTRACT_TYPE  AMT_ANNUITY  \
0           2030495      271877     Consumer loans     1730.430
1           2802425      108129         Cash loans    25188.615
2           2523466      122040         Cash loans    15060.735
3           2819243      176158         Cash loans    47041.335
4           1784265      202054         Cash loans    31924.395
...             ...         ...                ...          ...
1670209     2300464      352015     Consumer loans    14704.290
1670210     2357031      334635     Consumer loans     6622.020
1670211     2659632      249544     Consumer loans    11520.855
1670212     2785582      400317         Cash loans    18821.520
1670213     2418762      261212         Cash loans    16431.300

         AMT_APPLICATION  AMT_CREDIT  AMT_DOWN_PAYMENT
AMT_GOODS_PRICE  \
0                17145.0     17145.0              0.0
17145.0
1               607500.0    679671.0              NaN
607500.0
2               112500.0    136444.5              NaN
112500.0
3               450000.0    470790.0              NaN
450000.0
4               337500.0    404055.0              NaN
337500.0
...                  ...         ...              ...
...
1670209         267295.5    311400.0              0.0
267295.5
1670210          87750.0     64291.5          29250.0
```

```
2                        TUESDAY                         11  ...
3                         MONDAY                          7  ...
4                       THURSDAY                          9  ...
...                          ...                        ...  ...
1670209                 WEDNESDAY                         12  ...
1670210                   TUESDAY                         15  ...
1670211                    MONDAY                         12  ...
1670212                 WEDNESDAY                          9  ...
1670213                    SUNDAY                         10  ...

         NAME_SELLER_INDUSTRY  CNT_PAYMENT  NAME_YIELD_GROUP  \
0                Connectivity         12.0            middle
1                         XNA         36.0       low_action
2                         XNA         12.0              high
3                         XNA         12.0            middle
4                         XNA         24.0              high
...                       ...          ...               ...
1670209             Furniture         30.0       low_normal
1670210             Furniture         12.0            middle
1670211  Consumer electronics         10.0       low_normal
1670212                   XNA         12.0       low_normal
1670213                   XNA         48.0            middle

                 PRODUCT_COMBINATION DAYS_FIRST_DRAWING
DAYS_FIRST_DUE  \
0           POS mobile with interest          365243.0           -
42.0
1                Cash X-Sell: low          365243.0          -
134.0
2               Cash X-Sell: high          365243.0          -
271.0
3             Cash X-Sell: middle          365243.0          -
482.0
4               Cash Street: high               NaN
NaN
...                           ...               ...
...
1670209   POS industry with interest          365243.0          -
508.0
1670210   POS industry with interest          365243.0         -
1604.0
1670211  POS household with interest          365243.0         -
1457.0
1670212            Cash X-Sell: low          365243.0         -
1155.0
1670213         Cash X-Sell: middle          365243.0         -
1163.0

        DAYS_LAST_DUE_1ST_VERSION DAYS_LAST_DUE DAYS_TERMINATION  \
```

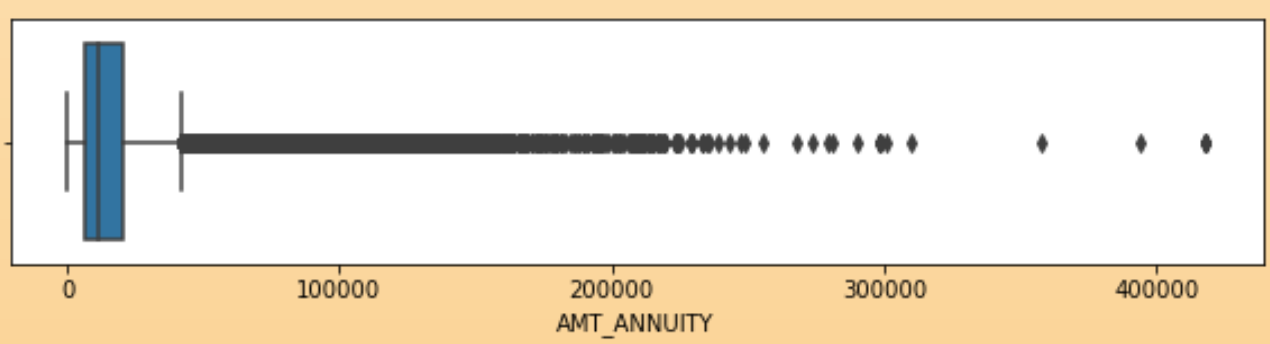
```
        NFLAG_INSURED_ON_APPROVAL
0                             0.0
1                             1.0
2                             1.0
3                             1.0
4                             NaN
...                           ...
1670209                       0.0
1670210                       0.0
1670211                       0.0
1670212                       1.0
1670213                       0.0

[1669867 rows x 35 columns]
```

B) Outliers Analysis

```
plt.figure(figsize=(10,2))
sns.boxplot(prev_ap_df['AMT_DOWN_PAYMENT'])
plt.show()
```

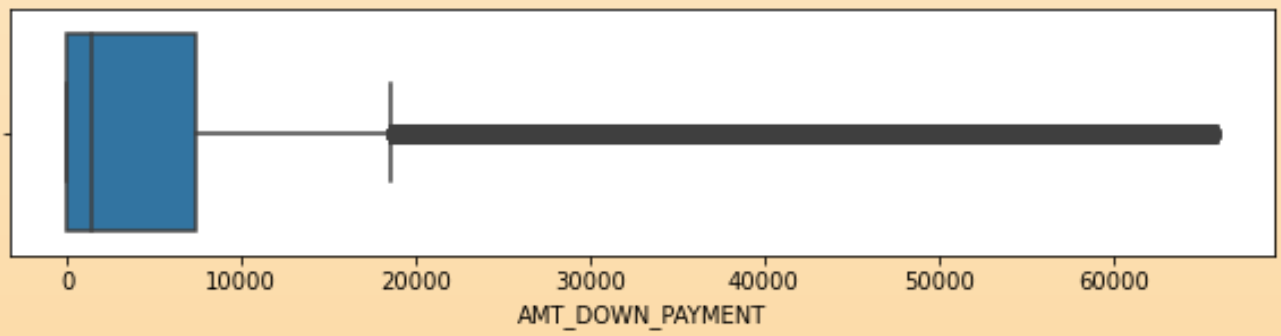



```
plt.figure(figsize=(10,2))
sns.boxplot(prev_ap_df['AMT_ANNUITY'])
plt.show()
```

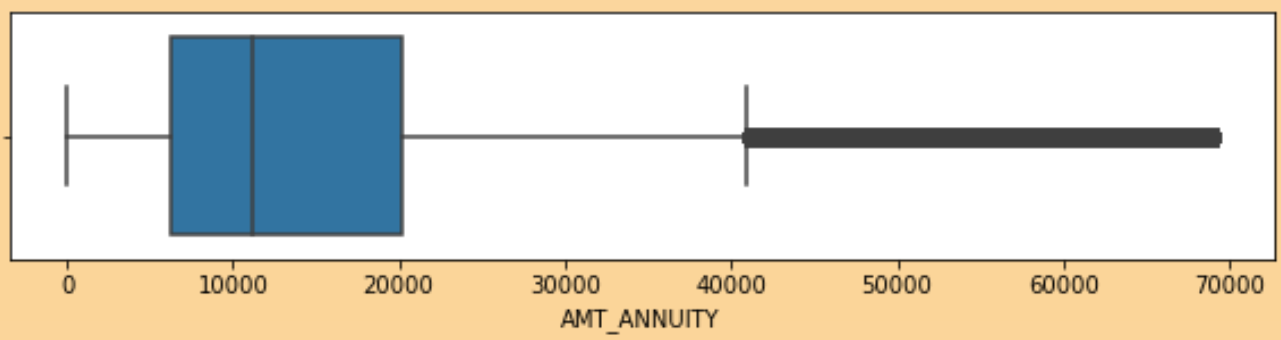



```
import pandas as pd

def remove_outliers(df, column):
    """
    Removes outliers outside of the 99th percentile in a Pandas DataFrame
for a given column.
```

```
    Parameters:
        df (Pandas DataFrame): The DataFrame containing the data.
        column (str): The name of the column to remove outliers for.

    Returns:
        The DataFrame with outliers removed.
    """

    # calculate the 99th percentile
    perc_99 = df[column].quantile(0.99)


    # remove outliers outside the IQR
    df = df[~(df[column] > perc_99)]

    return df

prev_ap_df = remove_outliers(prev_ap_df, 'AMT_DOWN_PAYMENT')

plt.figure(figsize=(10,2))
sns.boxplot(prev_ap_df['AMT_DOWN_PAYMENT'])
plt.show()
```




```
prev_ap_df = remove_outliers(prev_ap_df, 'AMT_ANNUITY')

plt.figure(figsize=(10,2))
sns.boxplot(prev_ap_df['AMT_ANNUITY'])
plt.show()
```



Now, I will try to perform data cleaning on the 'application_df'

a) Missing value handling

i) Identifying Missing Data

```
# We will use missingdata_percentage() function that we created earlier
missingdata_application_df = missingdata_percentage(application_df)

missingdata_application_df.sort_values('percentage', ascending=False)
```

```
                    category  percentage
41           COMMONAREA_MEDI   69.872297
13            COMMONAREA_AVG   69.872297
27           COMMONAREA_MODE   69.872297
49  NONLIVINGAPARTMENTS_MEDI   69.432963
35  NONLIVINGAPARTMENTS_MODE   69.432963
..                       ...         ...
7               EXT_SOURCE_2    0.214626
1            AMT_GOODS_PRICE    0.090403
0                AMT_ANNUITY    0.003902
5            CNT_FAM_MEMBERS    0.000650
60    DAYS_LAST_PHONE_CHANGE    0.000325

[67 rows x 2 columns]
```

```
missingdata_application_df[missingdata_application_df['percentage']<1]
```

```
                    category  percentage
0                AMT_ANNUITY    0.003902
1            AMT_GOODS_PRICE    0.090403
2            NAME_TYPE_SUITE    0.420148
5            CNT_FAM_MEMBERS    0.000650
7               EXT_SOURCE_2    0.214626
56  OBS_30_CNT_SOCIAL_CIRCLE    0.332021
57  DEF_30_CNT_SOCIAL_CIRCLE    0.332021
58  OBS_60_CNT_SOCIAL_CIRCLE    0.332021
59  DEF_60_CNT_SOCIAL_CIRCLE    0.332021
60    DAYS_LAST_PHONE_CHANGE    0.000325
```

Insights

There are 16 features in prev_app_df that have missing values.

ii) Dropping missing data

```
application_df.dropna(subset=['AMT_ANNUITY', 'AMT_GOODS_PRICE',
'NAME_TYPE_SUITE', 'CNT_FAM_MEMBERS', 'EXT_SOURCE_2',
'OBS_30_CNT_SOCIAL_CIRCLE','DEF_30_CNT_SOCIAL_CIRCLE',
'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_CIRCLE',
'DAYS_LAST_PHONE_CHANGE'], inplace=True)
```

```
#Checking the remaining columns
application_df
```

```
        SK_ID_CURR  TARGET NAME_CONTRACT_TYPE CODE_GENDER
FLAG_OWN_CAR  \
0           100002       1         Cash loans           M
N
1           100003       0         Cash loans           F
N
2           100004       0    Revolving loans           M
Y
3           100006       0         Cash loans           F
N
4           100007       0         Cash loans           M
N
...            ...     ...                ...         ...
...
307506      456251       0         Cash loans           M
N
307507      456252       0         Cash loans           F
N
307508      456253       0         Cash loans           F
N
307509      456254       1         Cash loans           F
N
307510      456255       0         Cash loans           F
N

       FLAG_OWN_REALTY  CNT_CHILDREN  AMT_INCOME_TOTAL  AMT_CREDIT
\
0                    Y             0          202500.0    406597.5
1                    N             0          270000.0   1293502.5
2                    Y             0           67500.0    135000.0
3                    Y             0          135000.0    312682.5
4                    Y             0          121500.0    513000.0
...                ...           ...               ...         ...
307506               N             0          157500.0    254700.0
307507               Y             0           72000.0    269550.0
307508               Y             0          153000.0    677664.0
307509               Y             0          171000.0    370107.0
307510               N             0          157500.0    675000.0

        AMT_ANNUITY  ...  FLAG_DOCUMENT_18 FLAG_DOCUMENT_19
FLAG_DOCUMENT_20  \
0           24700.5  ...                 0                0
0
1           35698.5  ...                 0                0
0
2            6750.0  ...                 0                0
0
```

```
4                     0                         0.0                         0
.0
...                 ...                         ...                         .
..
307506                0                         NaN                         N
aN
307507                0                         NaN                         N
aN
307508                0                         1.0                         0
.0
307509                0                         0.0                         0
.0
307510                0                         0.0                         0
.0

        AMT_REQ_CREDIT_BUREAU_WEEK  AMT_REQ_CREDIT_BUREAU_MON  \
0                              0.0                         0.0
1                              0.0                         0.0
2                              0.0                         0.0
3                              NaN                         NaN
4                              0.0                         0.0
...                            ...                         ...
307506                         NaN                         NaN
307507                         NaN                         NaN
307508                         0.0                         1.0
307509                         0.0                         0.0
307510                         0.0                         2.0

        AMT_REQ_CREDIT_BUREAU_QRT  AMT_REQ_CREDIT_BUREAU_YEAR
0                             0.0                         1.0
1                             0.0                         0.0
2                             0.0                         0.0
3                             NaN                         NaN
4                             0.0                         0.0
...                           ...                         ...
307506                        NaN                         NaN
307507                        NaN                         NaN
307508                        0.0                         1.0
307509                        0.0                         0.0
307510                        0.0                         1.0

[304531 rows x 122 columns]
```

c) Checking Data Imbalance

for checking imbalance in data we need to merge both dataframes and clean it jut as we have cleaned both the datasets separately

```
# Merging only required columns of application_data with previous_applicat
ion_data

prev_ap_merged = pd.merge(application_df[['SK_ID_CURR', 'TARGET']], prev_a
p_df, how='left', on=['SK_ID_CURR'])
print(prev_ap_merged.shape)
```

```
prev_ap_merged = remove_outliers(prev_ap_merged, 'AMT_DOWN_PAYMENT')
prev_ap_merged = remove_outliers(prev_ap_merged, 'AMT_ANNUITY')
print(prev_ap_merged.shape)
```

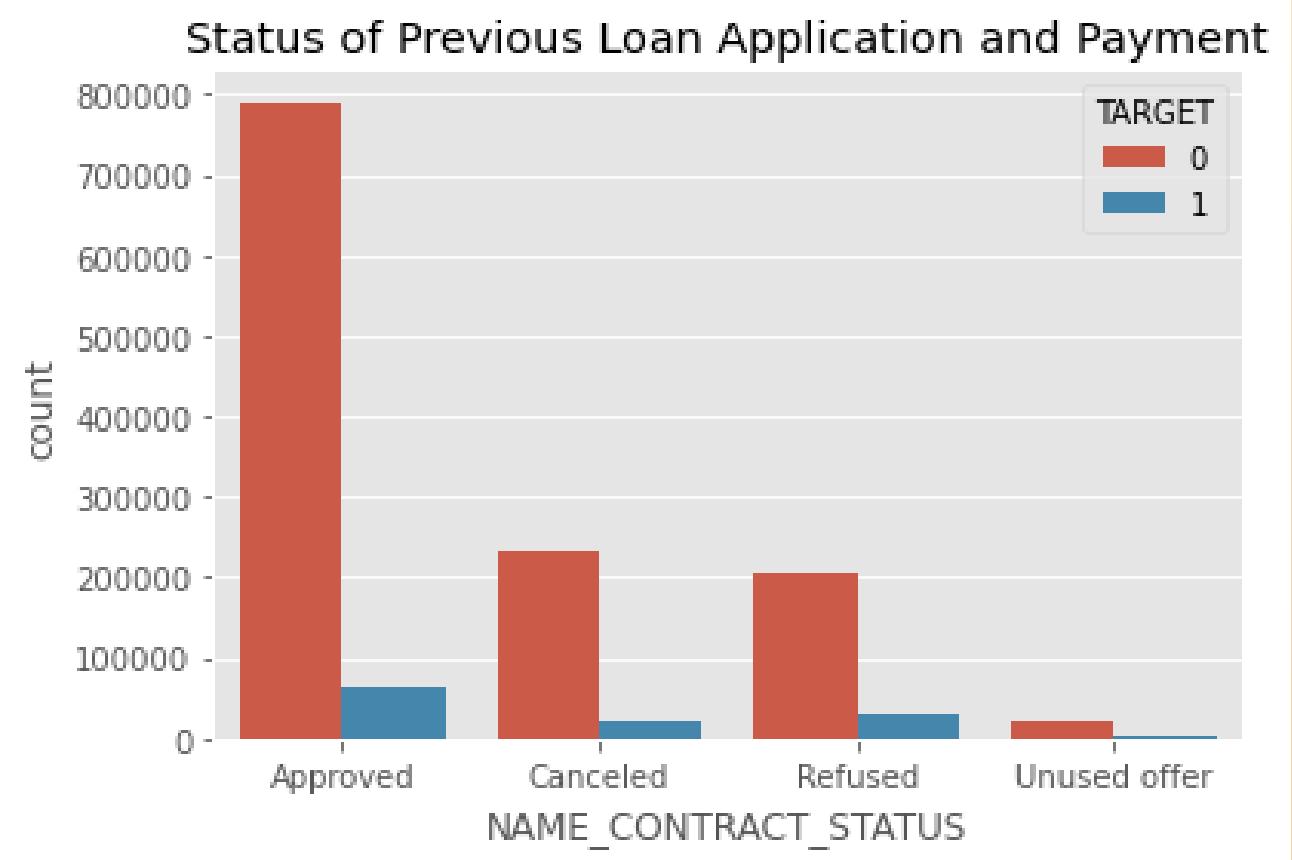
```
(1388267, 36)
```

```
prev_ap_merged.describe()
```

```
         SK_ID_CURR         TARGET     SK_ID_PREV    AMT_ANNUITY  \
count  1.388267e+06  1.388267e+06  1.371700e+06  1.066537e+06
mean   2.784320e+05  8.702505e-02  1.922904e+06  1.452031e+04
std    1.027883e+05  2.818719e-01  5.326533e+05  1.156755e+04
min    1.000020e+05  0.000000e+00  1.000001e+06  0.000000e+00
25%    1.893400e+05  0.000000e+00  1.461701e+06  6.126255e+03
50%    2.789110e+05  0.000000e+00  1.922842e+06  1.087384e+04
75%    3.674500e+05  0.000000e+00  2.384084e+06  1.939279e+04
max    4.562550e+05  1.000000e+00  2.845381e+06  5.805932e+04

       AMT_APPLICATION    AMT_CREDIT  AMT_DOWN_PAYMENT
AMT_GOODS_PRICE  \
count     1.371700e+06  1.371700e+06     646928.000000
1.054234e+06
mean      1.564674e+05  1.775130e+05       5071.408513
2.036190e+05
std       2.481789e+05  2.741488e+05       7987.331198
2.656373e+05
min       0.000000e+00  0.000000e+00         -0.900000
0.000000e+00
25%       1.818000e+04  2.384100e+04          0.000000
4.855590e+04
50%       6.750000e+04  7.695000e+04       1539.000000
1.043100e+05
75%       1.777500e+05  1.978200e+05       7164.000000
2.250000e+05
max       3.511305e+06  3.511305e+06      45000.000000
3.511305e+06

       HOUR_APPR_PROCESS_START  NFLAG_LAST_APPL_IN_DAY
RATE_DOWN_PAYMENT  \
count              1.371700e+06           1.371700e+06
646928.000000
mean               1.246973e+01           9.963768e-01
0.075369
std                3.329384e+00           6.008424e-02
0.099251
min                0.000000e+00           0.000000e+00          -
0.000015
25%                1.000000e+01           1.000000e+00
0.000000
50%                1.200000e+01           1.000000e+00
```

```
mean      13859.406067                34283.481966   76815.867727
std       72536.846349               107580.829373  149819.845989
min       -2892.000000                -2801.000000   -2889.000000
25%       -1648.000000                -1261.000000   -1331.000000
50%        -835.000000                 -366.000000    -544.000000
75%        -408.000000                  136.000000     -74.000000
max      365243.000000               365243.000000  365243.000000

       DAYS_TERMINATION  NFLAG_INSURED_ON_APPROVAL
count     825220.000000               825220.000000
mean       82618.287087                    0.327315
std       153717.777294                    0.469234
min        -2874.000000                    0.000000
25%        -1284.000000                    0.000000
50%         -503.000000                    0.000000
75%          -41.000000                    1.000000
max       365243.000000                    1.000000
```

```
plt.style.use("ggplot")
plt.title("Status of Previous Loan Application and Payment")
sns.countplot(prev_ap_merged['NAME_CONTRACT_STATUS'], hue=prev_ap_merged['
TARGET'])
plt.show()
```




```
# Percentage of previously approved loan applicants that defaulted in
current loan
```

```
total_approved = prev_ap_merged[prev_ap_merged['NAME_CONTRACT_STATUS'] ==
"Approved"].shape[0]
default_approved = prev_ap_merged[(prev_ap_merged['TARGET'] == 1) & (prev_
ap_merged['NAME_CONTRACT_STATUS'] == "Approved")].shape[0]

print("Percentage of previously approved loan applicants that defaulted in
current loan : {}".format(default_approved/total_approved*100))
```

```
Percentage of previously approved loan applicants that defaulted in curren
t loan : 7.678348657899435
```

```
total_refused = prev_ap_merged[prev_ap_merged['NAME_CONTRACT_STATUS'] == "
Refused"].shape[0]
nondefault_refused = prev_ap_merged[(prev_ap_merged['TARGET'] == 0) & (pre
v_ap_merged['NAME_CONTRACT_STATUS'] == "Refused")].shape[0]

print("Percentage of previously refused loan applicants that were able to
pay current loan : {}".format(nondefault_refused/total_refused*100))
```

```
Percentage of previously refused loan applicants that were able to pay cur
rent loan : 87.86236584753729
```

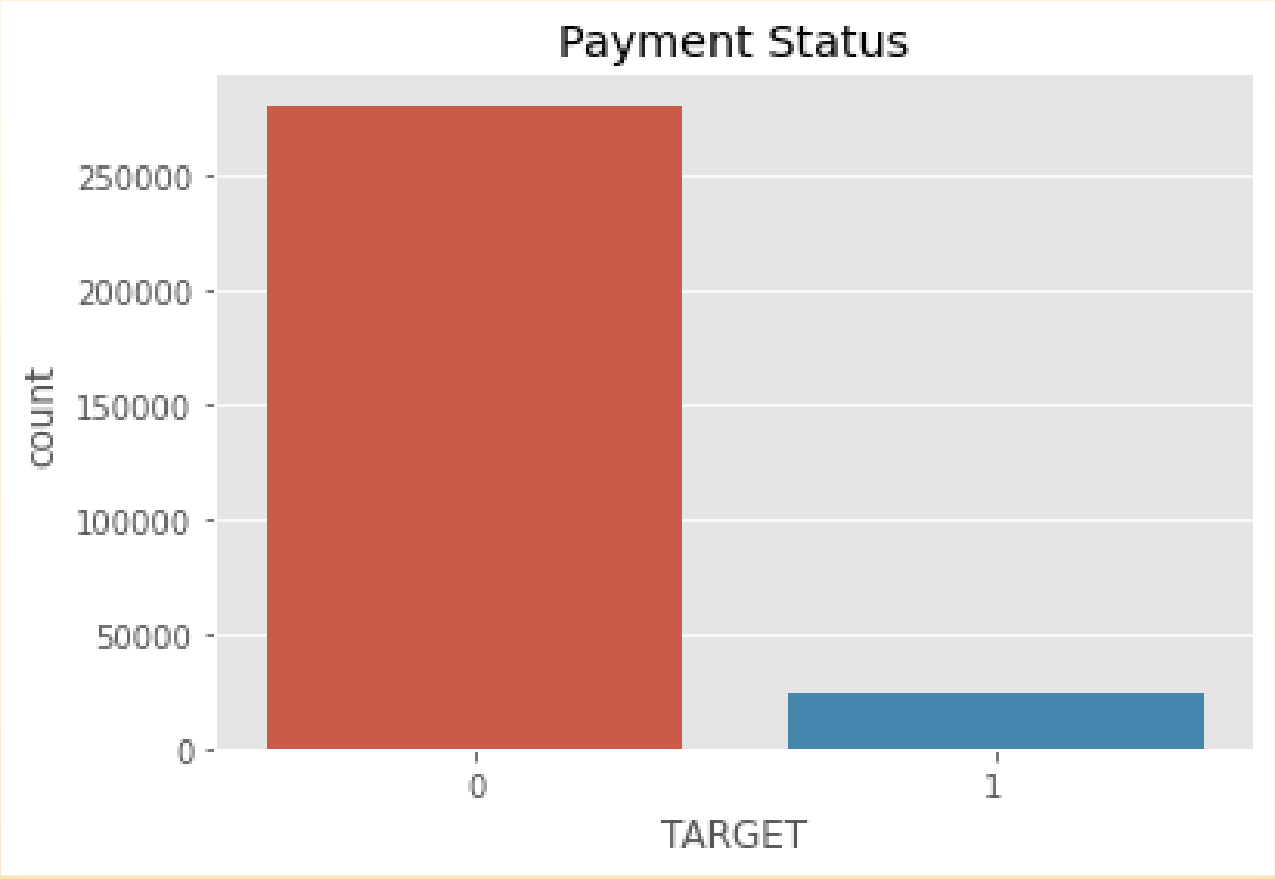
Insights :

The applicants whose previous loans were approved are more likely to pay current loan in time, than the applicants whose previous loans were rejected.

7.6% of the previously approved loan applicants that defaulted in current loan

87.8 % of the previously refused loan applicants that were able to pay current loan

```
plt.title("Payment Status")
sns.countplot(application_df['TARGET'])
plt.show()
```



This data is highly imbalanced as number of defaulter is very less in total population.

```
non_default  = application_df[application_df["TARGET"] == 0]
default  = application_df[application_df["TARGET"] == 1]

print("No. of defaulters: ", default.shape[0])
print("No. of non-defaulters: ", non_default.shape[0])
```

```
No. of defaulters:  24667
No. of non-defaulters:  279864
```

```
print("Percentage of defaulters: ", default.shape[0]*100/(default.shape[0]
+non_default.shape[0]))
```

```
Percentage of defaulters:  8.099996387888261
```

Insights :

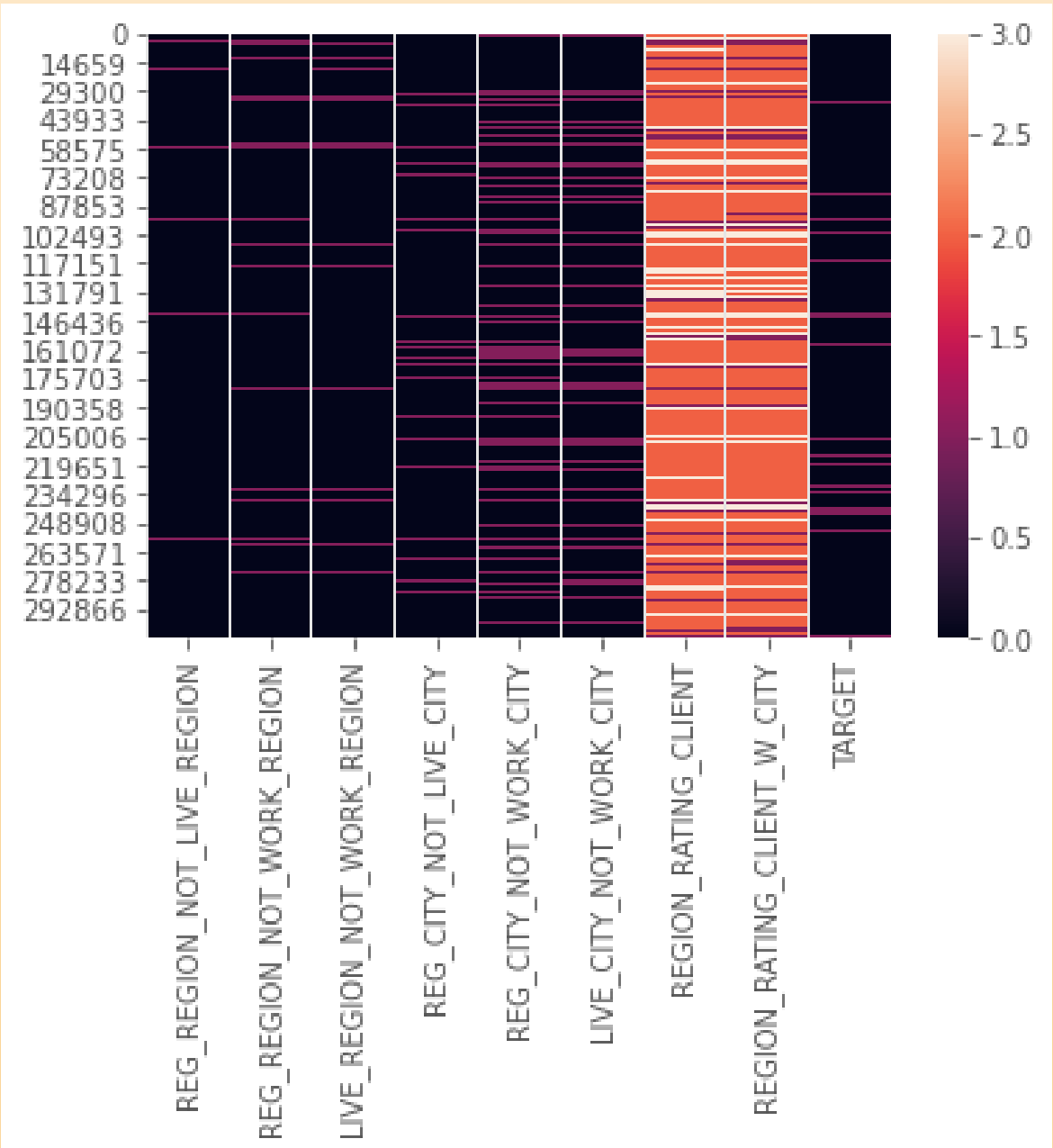
This data is highly imbalanced as number of defaulter is very less in total population.
Data Imbalance Ratio

Defaulter : Non-Defaulter = 8 : 92 = 2 : 23

4) Analysis of diffrent variables in segments

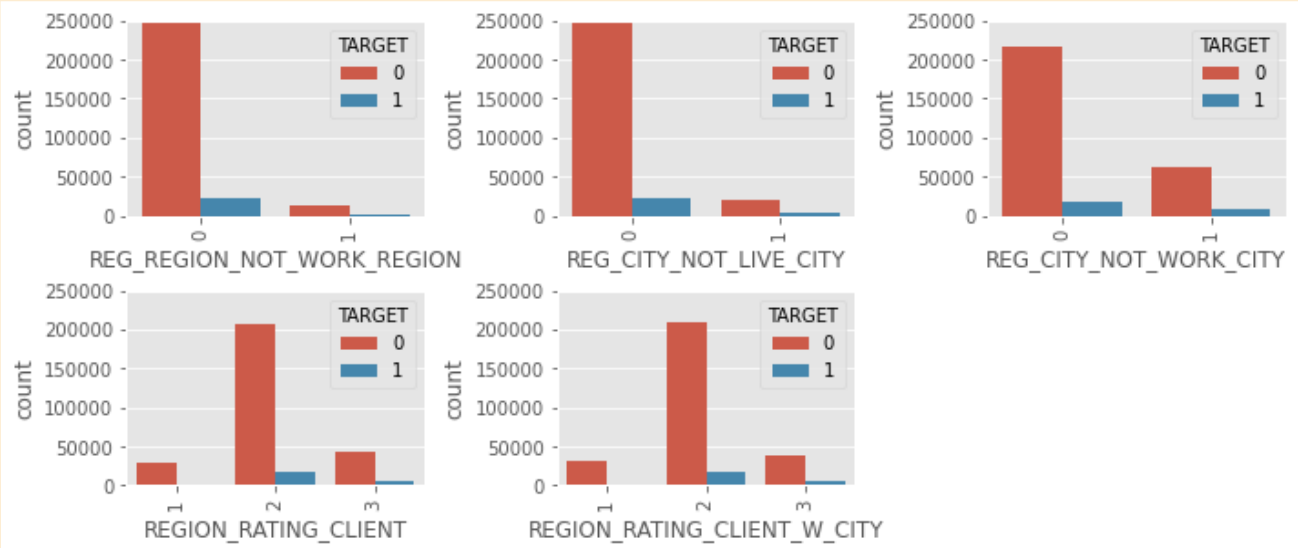
Segment 1 :- Region based analysis

```
start_idx = application_df.columns.get_loc('REG_REGION_NOT_LIVE_REGION')
end_idx = application_df.columns.get_loc('LIVE_CITY_NOT_WORK_CITY')

region_df = application_df.iloc[:, start_idx:end_idx+1]

region_df['REGION_RATING_CLIENT'] = application_df['REGION_RATING_CLIENT']
region_df['REGION_RATING_CLIENT_W_CITY'] = application_df['REGION_RATING_C
LIENT_W_CITY']
region_df["TARGET"] = application_df["TARGET"]

sns.heatmap(region_df)
plt.show()
```



Insights:

```
fig=plt.subplots(figsize=(10, 10))

for i, j in enumerate(['REG_REGION_NOT_WORK_REGION','REG_CITY_NOT_LIVE_CIT
Y', 'REG_CITY_NOT_WORK_CITY','REGION_RATING_CLIENT', 'REGION_RATING_CLIENT
_W_CITY']):
    plt.subplot(5, 3, i+1, ylim=(0, 250000))
    plt.subplots_adjust(hspace = 1.0)
    sns.countplot(application_df[j], hue=application_df["TARGET"])
    plt.xticks(rotation=90)
    plt.tight_layout()
```



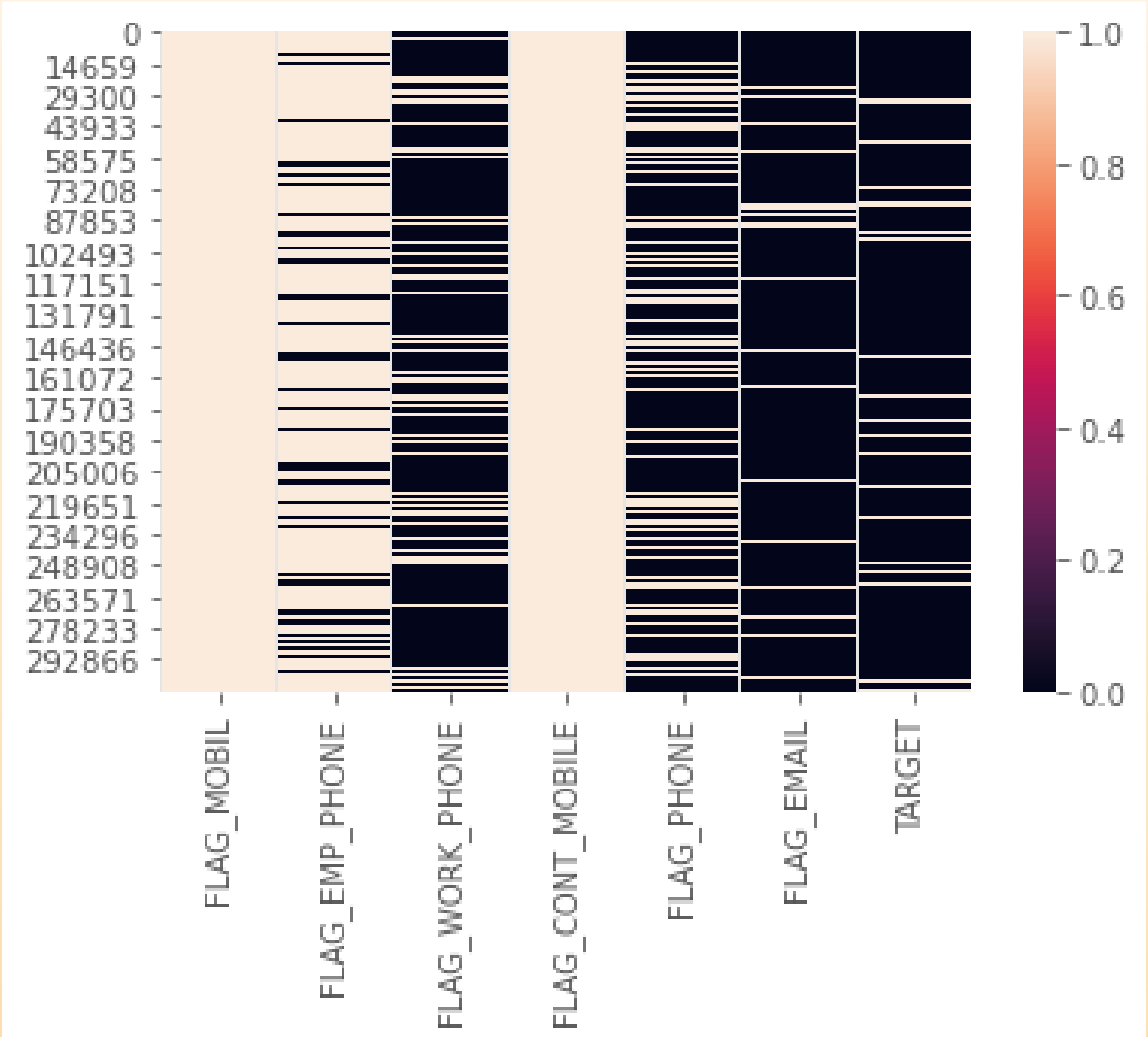
Insights:

Segment 2 :- Based on contact

```
contact_df = application_df[['FLAG_MOBIL','FLAG_EMP_PHONE','FLAG_WORK_PHON
E','FLAG_CONT_MOBILE','FLAG_PHONE','FLAG_EMAIL','DAYS_LAST_PHONE_CHANGE',
'TARGET']]
contact_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 304531 entries, 0 to 307510
Data columns (total 8 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   FLAG_MOBIL              304531 non-null  int64
 1   FLAG_EMP_PHONE          304531 non-null  int64
 2   FLAG_WORK_PHONE         304531 non-null  int64
 3   FLAG_CONT_MOBILE        304531 non-null  int64
 4   FLAG_PHONE              304531 non-null  int64
 5   FLAG_EMAIL              304531 non-null  int64
 6   DAYS_LAST_PHONE_CHANGE  304531 non-null  float64
 7   TARGET                  304531 non-null  int64
dtypes: float64(1), int64(7)
memory usage: 29.0 MB
```

```
plt.figure()
sns.heatmap(contact_df.drop('DAYS_LAST_PHONE_CHANGE', axis=1))
plt.show()
```



Insights:

As there is no similarity of patterns of TARGET value with the features,

Hence all of these features can be removed.

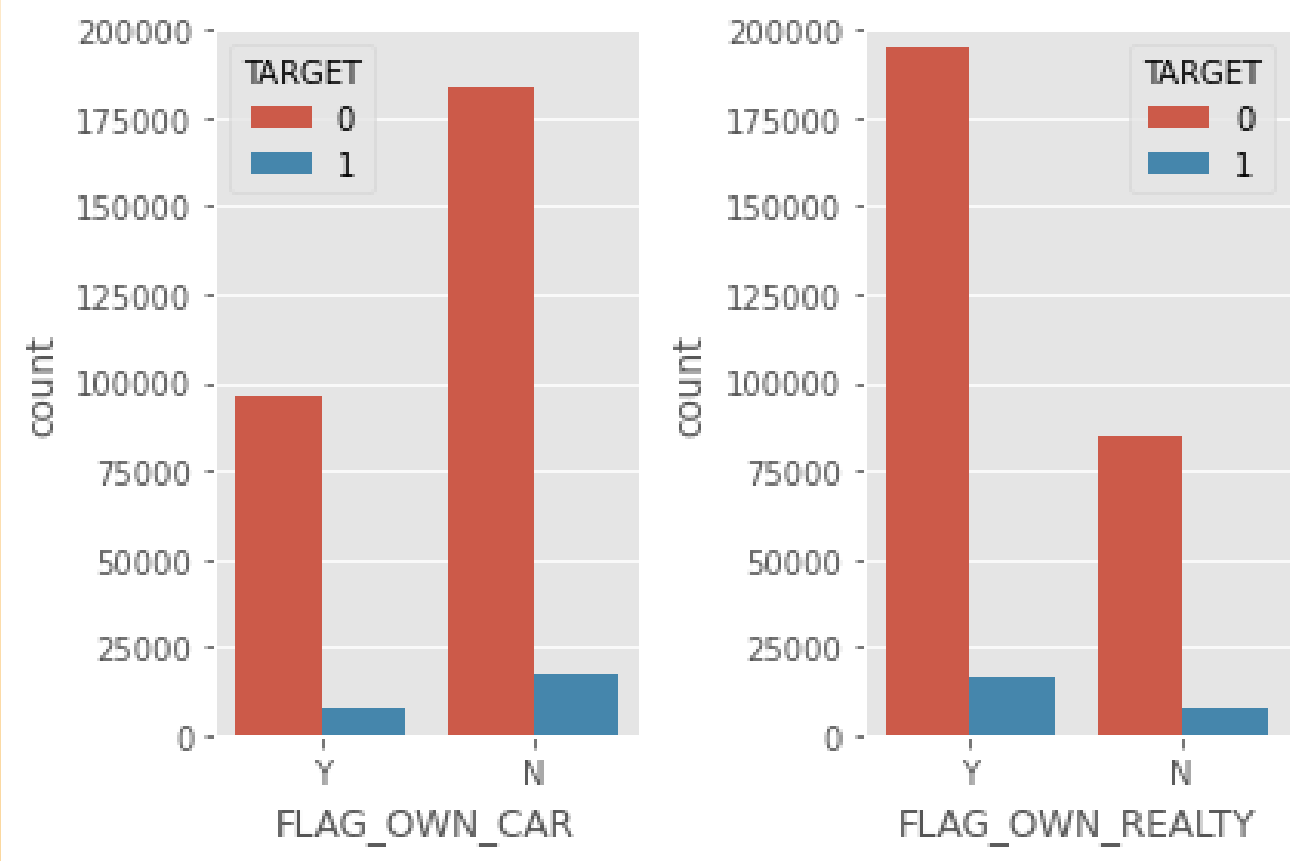
Segment 3 :- Based on assets owned

```
application_df[['FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'OWN_CAR_AGE', 'TARGET'
]].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 304531 entries, 0 to 307510
Data columns (total 4 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   FLAG_OWN_CAR     304531 non-null  object
 1   FLAG_OWN_REALTY  304531 non-null  object
 2   OWN_CAR_AGE      103619 non-null  float64
 3   TARGET           304531 non-null  int64
```

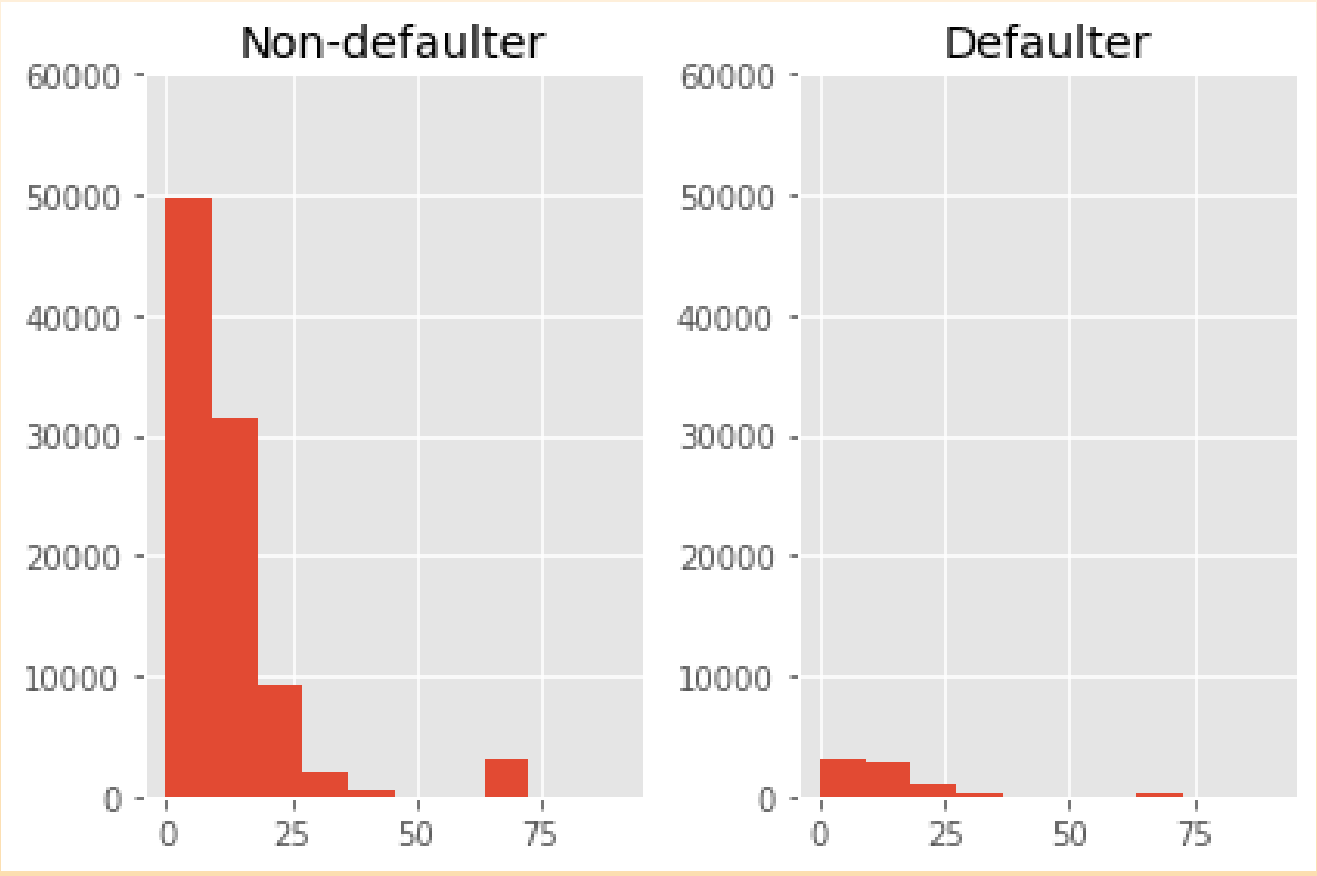
```
dtypes: float64(1), int64(1), object(2)
memory usage: 19.7+ MB
```

```
fig = plt.figure()

ax1 = fig.add_subplot(1, 2, 1, ylim=(0,200000))
ax2 = fig.add_subplot(1, 2, 2, ylim=(0,200000))

sns.countplot(application_df['FLAG_OWN_CAR'], hue=application_df['TARGET']
, order=['Y','N'], ax=ax1)
sns.countplot(application_df['FLAG_OWN_REALTY'], hue=application_df['TARGE
T'], order=['Y','N'], ax=ax2)

plt.tight_layout()

plt.show()
```




```
fig = plt.figure()

ax1 = fig.add_subplot(1, 2, 1, ylim=(0,60000), title="Non-defaulter")
ax2 = fig.add_subplot(1, 2, 2, ylim=(0,60000), title="Defaulter")

non_default['OWN_CAR_AGE'].hist(bins=10, ax=ax1)
default['OWN_CAR_AGE'].hist(bins=10, ax=ax2)

plt.tight_layout()
```

```
plt.show()
```

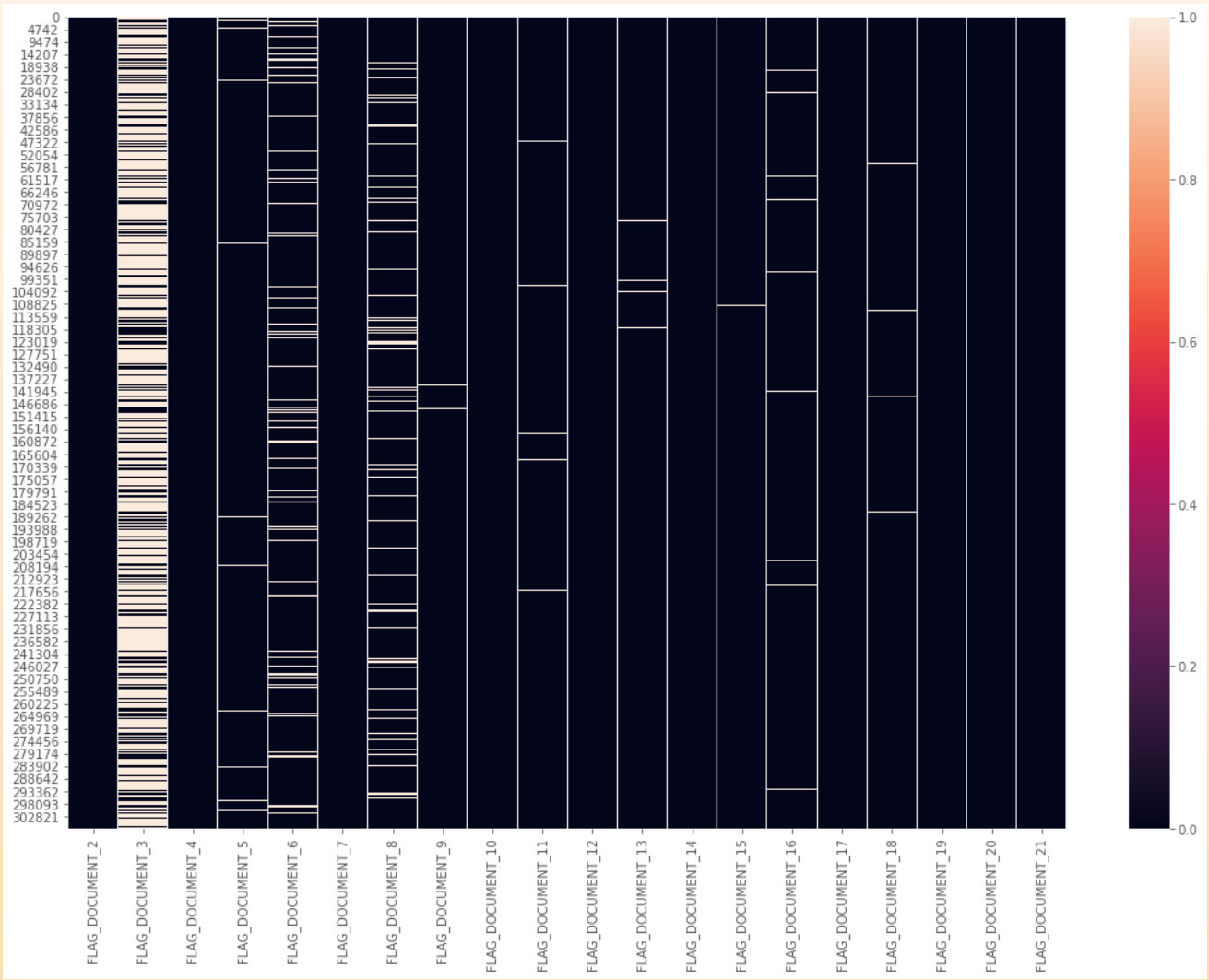


Insights :-

Most people between the age of 0 to 25 have cars irrespective of weather they are defaulter or not.
 both target values are similar so this feature can also be dropped.
People not owning reality and car have a slightly higher default rate than the people who own reality and car.

Segement 4 :- Based on documents submitted

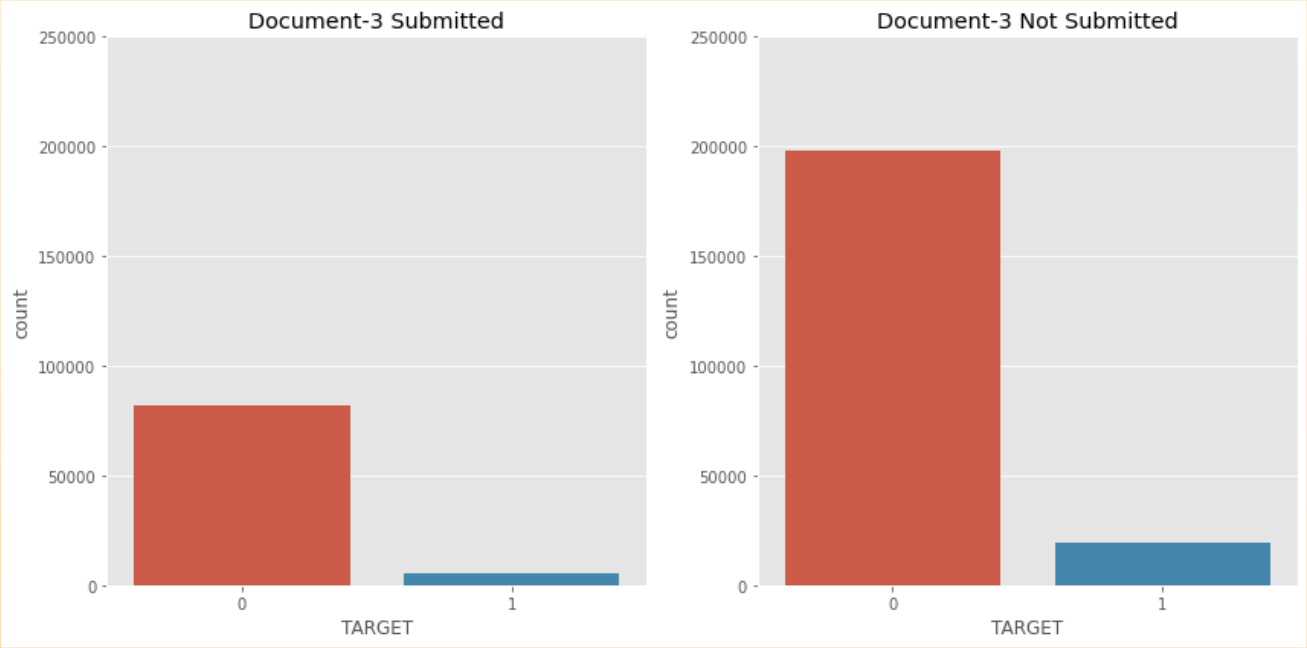
```
starting_idx = application_df.columns.get_loc("FLAG_DOCUMENT_2")
ending_idx = application_df.columns.get_loc("FLAG_DOCUMENT_21")+1

plt.figure(figsize=(18,12))
sns.heatmap(application_df.iloc[:,starting_idx:ending_idx])
plt.show()
```



Insights :-

We will now reconfirm the importance of document 3.

```
fig = plt.figure(figsize=(12,6))

ax1 = fig.add_subplot(1, 2, 1, ylim=(0,250000), title="Document-3 Submitte
d")
ax2 = fig.add_subplot(1, 2, 2, ylim=(0,250000), title="Document-3 Not Subm
itted")

sns.countplot(application_df[application_df["FLAG_DOCUMENT_3"] == 0]["TARG
ET"], ax=ax1)
sns.countplot(application_df[application_df["FLAG_DOCUMENT_3"] == 1]["TARG
ET"], ax=ax2)

plt.tight_layout()

plt.show()
```

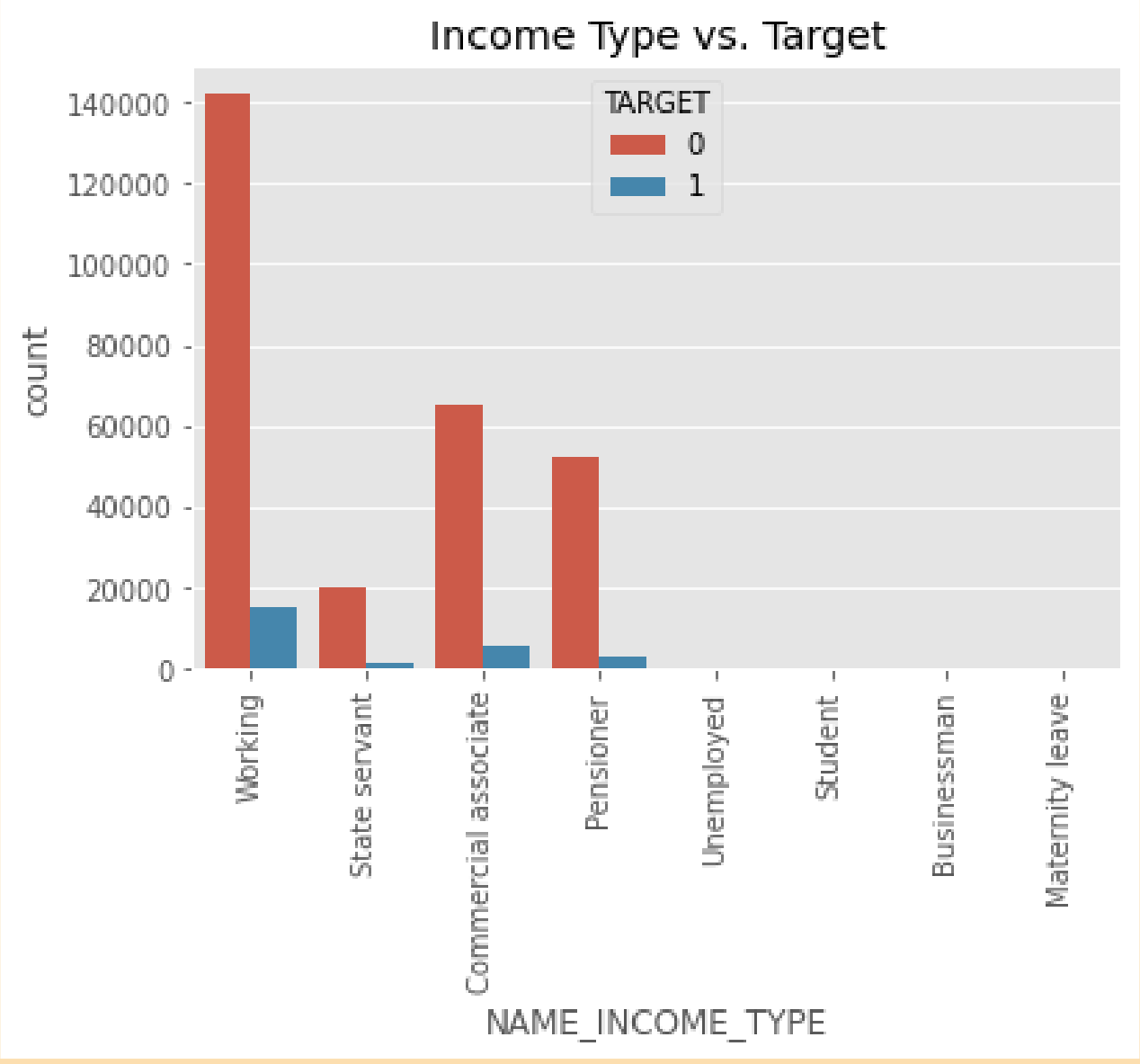


Lookig at the bar graph above, document 3 is shpwing similar trends for both defaulters and non-defaulters.

Hence we can drop this column.

Segment 5 :- Based on education level or occupation

```
plt.figure()
sns.countplot(application_df['NAME_INCOME_TYPE'], hue=application_df["TARGET"])
plt.xticks(rotation=90)
plt.title("Income Type vs. Target")
plt.show()
```




```
def value_wise_defaulter_percentage(df, col):
    new_df = pd.DataFrame(columns=['Value', 'Percentage of Defaulter'])

    for value in df[col].unique():
        default_cnt = df[(df[col] == value) & (df.TARGET == 1)].shape[0]
        total_cnt = df[df[col] == value].shape[0]
        new_df = new_df.append({'Value' : value , 'Percentage of Defaulter
' : (default_cnt*100/total_cnt)}, ignore_index=True)
    return new_df.sort_values(by='Percentage of Defaulter', ascending=Fals
e)

value_wise_defaulter_percentage(application_df,'NAME_INCOME_TYPE')
```

	Value	Percentage of Defaulter
4	Unemployed	42.105263
7	Maternity leave	40.000000
0	Working	9.620506
2	Commercial associate	7.517586
1	State servant	5.761719
3	Pensioner	5.395598

```
5               Student                 0.000000
6          Businessman                  0.000000
```

```
value_wise_defaulter_percentage(application_df,'NAME_EDUCATION_TYPE')
```

```
                           Value  Percentage of Defaulter
3                Lower secondary                10.897098
0  Secondary / secondary special                 8.963349
2              Incomplete higher                 8.501229
1               Higher education                 5.381937
4                Academic degree                 1.840491
```

```
value_wise_defaulter_percentage(application_df, 'OCCUPATION_TYPE')
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last
)
<ipython-input-51-c368d444f237> in <module>
----> 1 value_wise_defaulter_percentage(application_df, 'OCCUPATION_TYPE')

<ipython-input-46-83dad9d94a02> in value_wise_defaulter_percentage(df, col
)
      5         default_cnt = df[(df[col] == value) & (df.TARGET == 1)].sh
ape[0]
      6         total_cnt = df[df[col] == value].shape[0]
----> 7         new_df = new_df.append({'Value' : value , 'Percentage of D
efaulter' : (default_cnt*100/total_cnt)}, ignore_index=True)
      8     return new_df.sort_values(by='Percentage of Defaulter', ascend
ing=False)

ZeroDivisionError: division by zero
```

```
application_df['OCCUPATION_TYPE'].isnull().value_counts()
```

```
False    209096
True      95435
Name: OCCUPATION_TYPE, dtype: int64
```

```
application_df['OCCUPATION_TYPE'].value_counts()
```

```
Laborers                 54730
Sales staff              31790
Core staff               27263
Managers                 21114
Drivers                  18456
High skill tech staff    11261
Accountants               9698
Medicine staff            8459
Security staff            6667
Cooking staff             5898
Cleaning staff            4615
Private service staff     2629
```

```
Waiters/barmen staff      1335
Secretaries               1293
Realty agents              742
HR staff                   558
IT staff                   511
Name: OCCUPATION_TYPE, dtype: int64
```

```
# Imputing missing value for OCCUPATION TYPE as "Unknown"

application_df['OCCUPATION_TYPE'].fillna("Unknown", inplace=True)

value_wise_defaulter_percentage(application_df, 'OCCUPATION_TYPE')
```

	Value	Percentage of Defaulter
14	Low-skill Laborers	17.284545
13	Waiters/barmen staff	11.385768
5	Drivers	11.367577
11	Security staff	10.754462
0	Laborers	10.608441
8	Cooking staff	10.512038
7	Cleaning staff	9.642470
6	Sales staff	9.635105
15	Realty agents	7.951482
16	Secretaries	7.115236
10	Medicine staff	6.691098
17	IT staff	6.653620
9	Private service staff	6.542412
4	Unknown	6.534290
1	Core staff	6.330925
3	Managers	6.251776
12	High skill tech staff	6.180623
18	HR staff	6.093190
2	Accountants	4.866983

```
sns.countplot(application_df['CODE_GENDER'], hue=application_df["TARGET"])
plt.show()
```


```
value_wise_defaulter_percentage(application_df, 'CODE_GENDER')
```

```
  Value  Percentage of Defaulter
0     M                10.191744
1     F                 7.014595
2   XNA                 0.000000
```

Insights :-

Applicants that are on maternity leave or unemployeed are mst likely t
o be defaulter.
Businessmen and studennts have lowest default rate (zero).
Applicants that have "Lower secondary" education are most likely to be
defaulter as compared to others.
Low skilled labourers have very high rate of defaulters.
 Although there are more female applicants but male aplicants are more
likely to be defaulter than famale applicants

5) Correlation

a) Top 5 correlation for defaulters

```
defaulter_corr = default.corr()
round(defaulter_corr, 2)

corr_list = defaulter_corr.unstack()

# Listing the correlations in pair sorted in descending order
corr_list.sort_values(ascending=False).drop_duplicates().head(6)
```

```
SK_ID_CURR                 SK_ID_CURR                   1.000000
OBS_30_CNT_SOCIAL_CIRCLE   OBS_60_CNT_SOCIAL_CIRCLE    0.998286
BASEMENTAREA_MEDI          BASEMENTAREA_AVG             0.998205
YEARS_BUILD_MEDI           YEARS_BUILD_AVG              0.998090
COMMONAREA_MEDI            COMMONAREA_AVG               0.998085
NONLIVINGAPARTMENTS_AVG    NONLIVINGAPARTMENTS_MEDI     0.998053
dtype: float64
```

Top 10 Correlations for Defaulters

b) Top 5 correlation for non-defaulters

```
nondefaulter_corr = non_default.corr()
round(nondefaulter_corr, 2)

nondf_corr_list = nondefaulter_corr.unstack()

# Listing the correlations in pair sorted in descending order
nondf_corr_list.sort_values(ascending=False).drop_duplicates().head(6)
```

```
SK_ID_CURR                 SK_ID_CURR                   1.000000
YEARS_BUILD_AVG            YEARS_BUILD_MEDI             0.998519
OBS_30_CNT_SOCIAL_CIRCLE   OBS_60_CNT_SOCIAL_CIRCLE    0.998513
FLOORSMIN_AVG              FLOORSMIN_MEDI               0.997215
FLOORSMAX_MEDI             FLOORSMAX_AVG                0.997032
ENTRANCES_MEDI             ENTRANCES_AVG                0.996910
dtype: float64
```

Top 10 Correlations for Non-Defaulters

6) Summary

- This data is highly imbalanced.
- Based on region : Defaulter rate is highest when permanent address and working address is same.
- Based on Contact : Considered 'FLAG_MOBIL','FLAG_EMP_PHONE' etc for this segment. No impact on Target, features can be dropped.
- Most of the applicants own realty, most of the applicants do not own cars, people not owning reality and car and have a slightly higher default rate than the people who own reality and car.
- Based on Documents : Majority of the applicants did not submit any documents apart from DOCUMENT_3. FLAG_DOCUMENT_3 has similar impact on defaulters and non-defaulters. Hence this column can be dropped.
- Applicants that are either low skilled labour or have attained lower secondary education or that are unemployed have higher rates of being a defaulter.

Module - 7

XYZ Ads Airing Report

Project Description :

For your Final Project, we are providing you with a dataset having different TV Airing Brands, their product, their category. Dataset includes the network through which Ads are airing, types of networks like Cable/ Broadcast and the show name also on which Ads got aired. You can also see the data of Dayparts, Time zone and the time & date at which Ads got aired. IT also includes other data like Pod Position (the lesser the valuable), duration for which Ads aired on screen, Equivalent sales &, total amount spent on the Ads aired.

Approach :

- Start with a clear and concise summary of your insights
- Use clear and concise language
- Use visual aids
- Provide context
- Highlight key findings
- Consider the audience

Tech Stack Used :

1. MS Excel

Link to excel File :

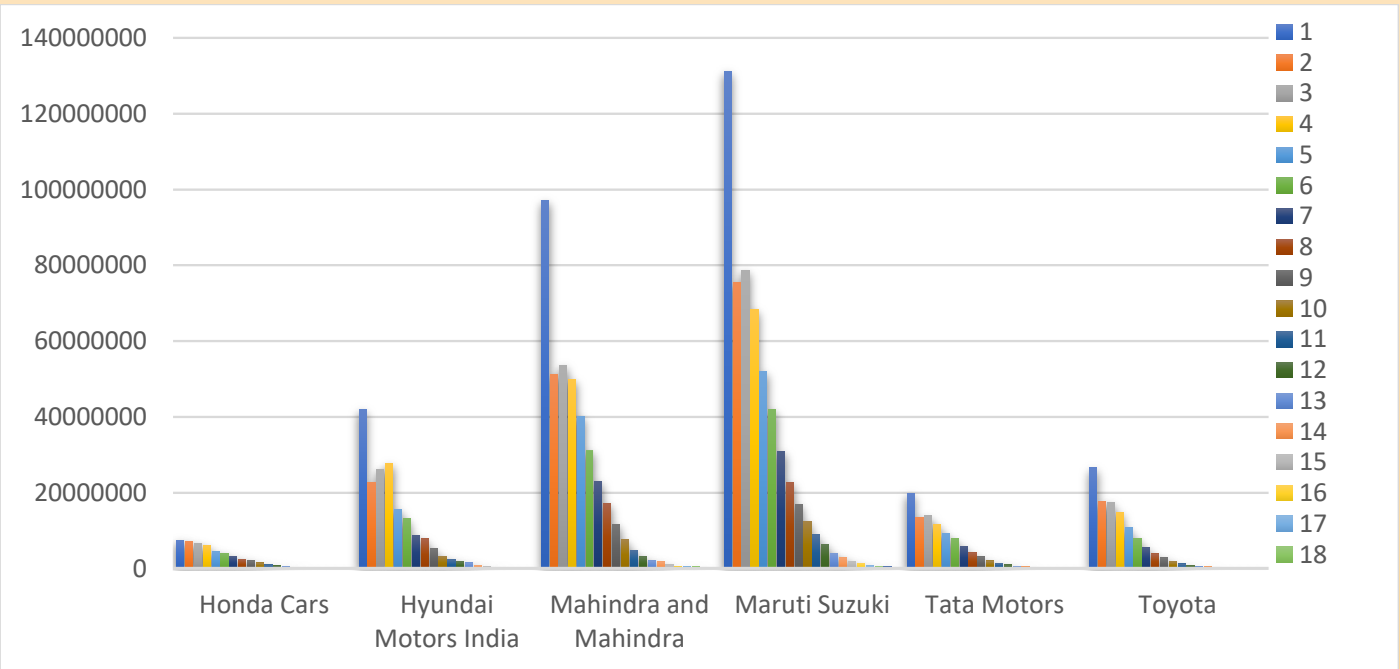
Click Here

Insights :

A. *What is Pod Position? Does the Pod position number affect the amount spent on Ads for a specific period of time by a company? (Explain in Details with examples from the dataset provided)*

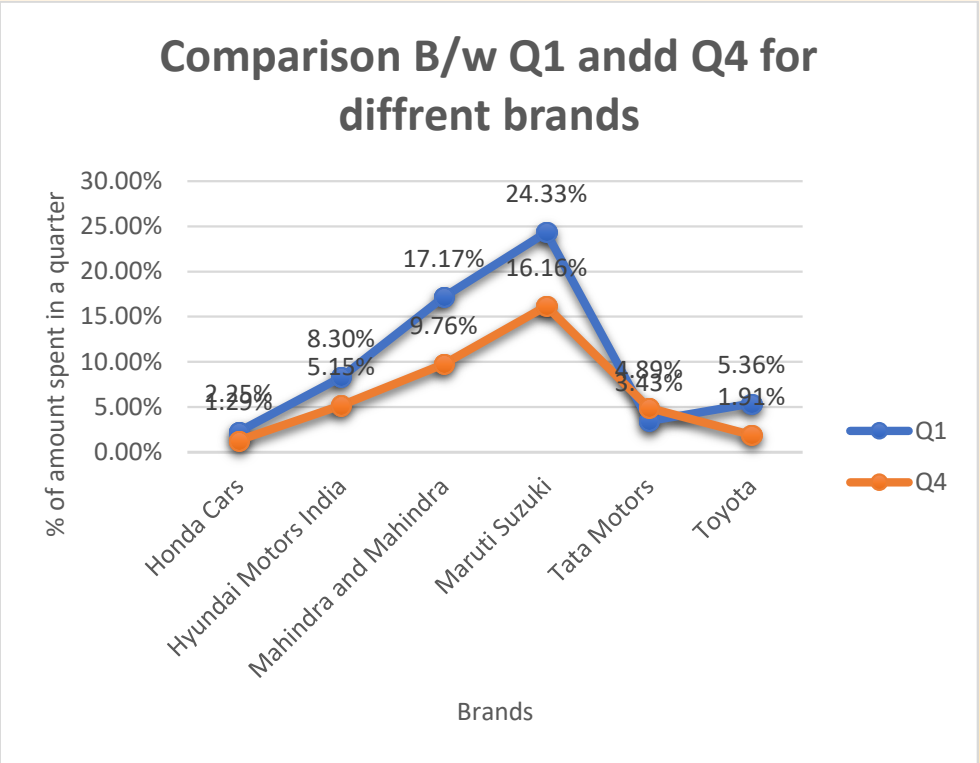
Pod Position refers to the placement of an advertisement within a pod or block of TV commercials. For example, if there are five ads airing during a commercial break, the pod position of a particular ad would be its rank within that group of five.

In general, pod position is believed to be an important factor in the effectiveness of TV advertising. Ads that air earlier in a pod may be more memorable and have a greater impact on viewers, while ads that air later may be more likely to be skipped or ignored.

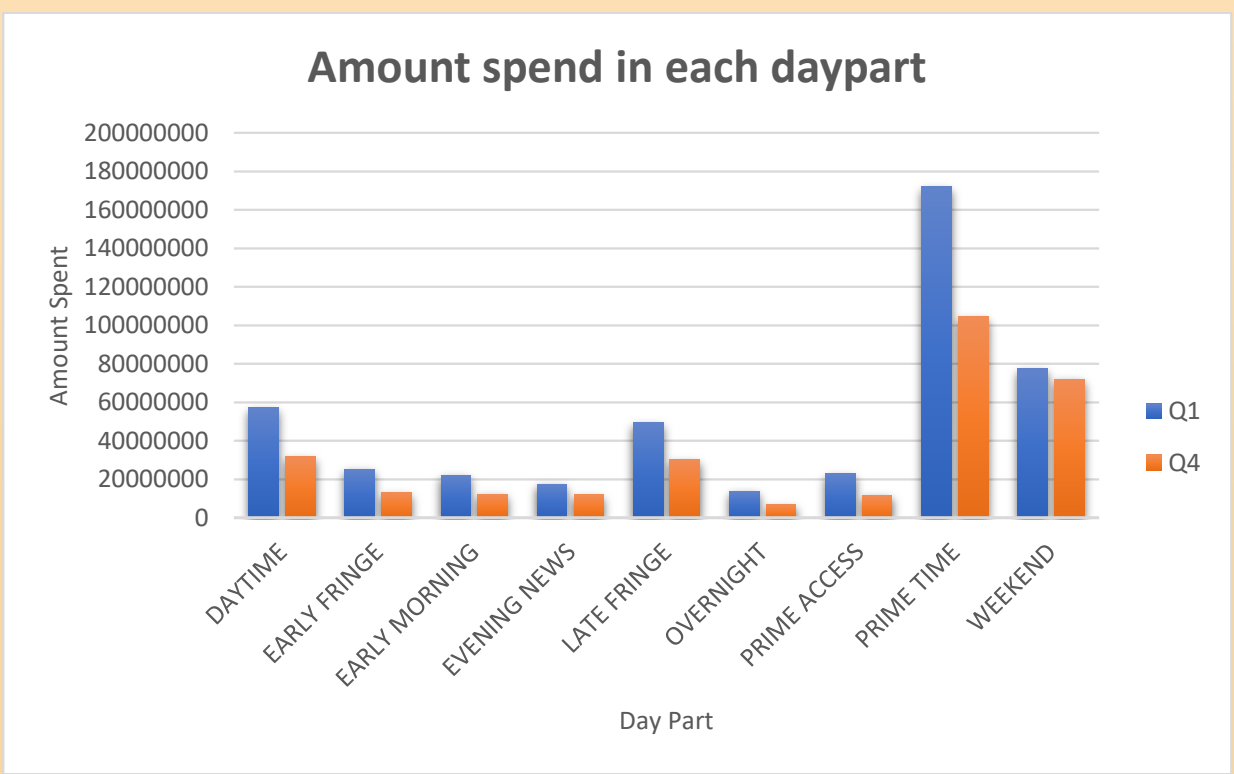


Analysis of the TV advertising dataset reveals that Maruti Suzuki spends the most on ads in position 1. Among all the ads that aired in position 1, those for Maruti Suzuki had the highest total amount spent, with a total of 131,181,074.

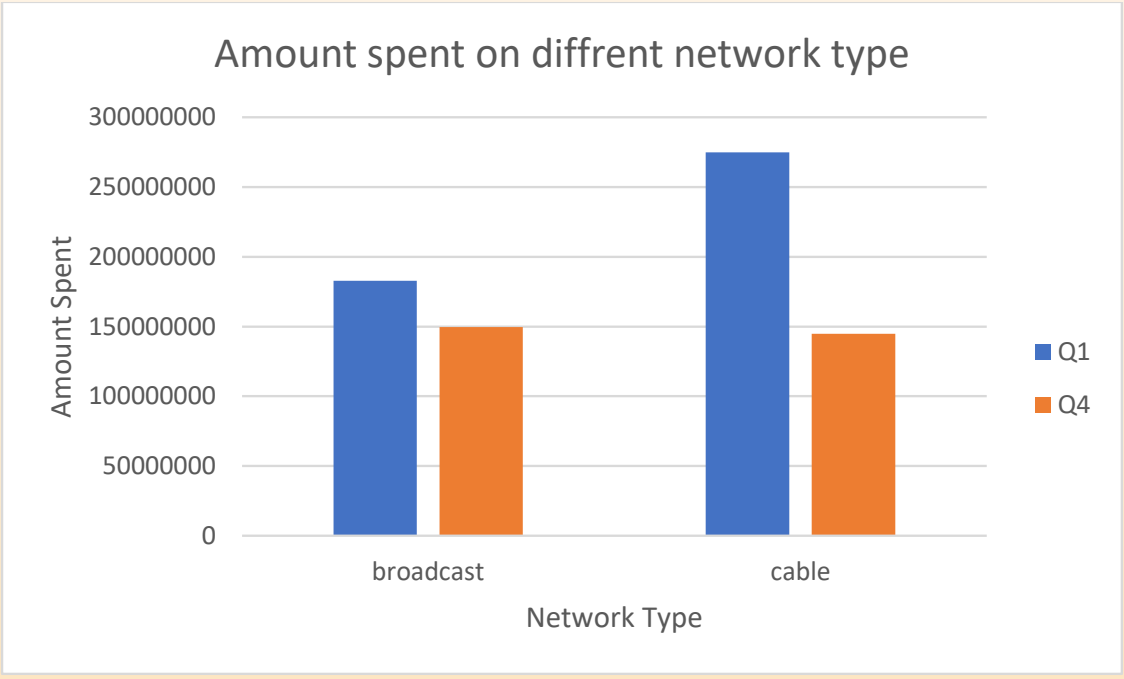
A. *What is the share of various brands in TV airings and how has it changed from Q1 to Q4 in 2021?*



Amount spent on ads in Q4 has been lower than that of Q4 by almost every brand except Tata Motors.



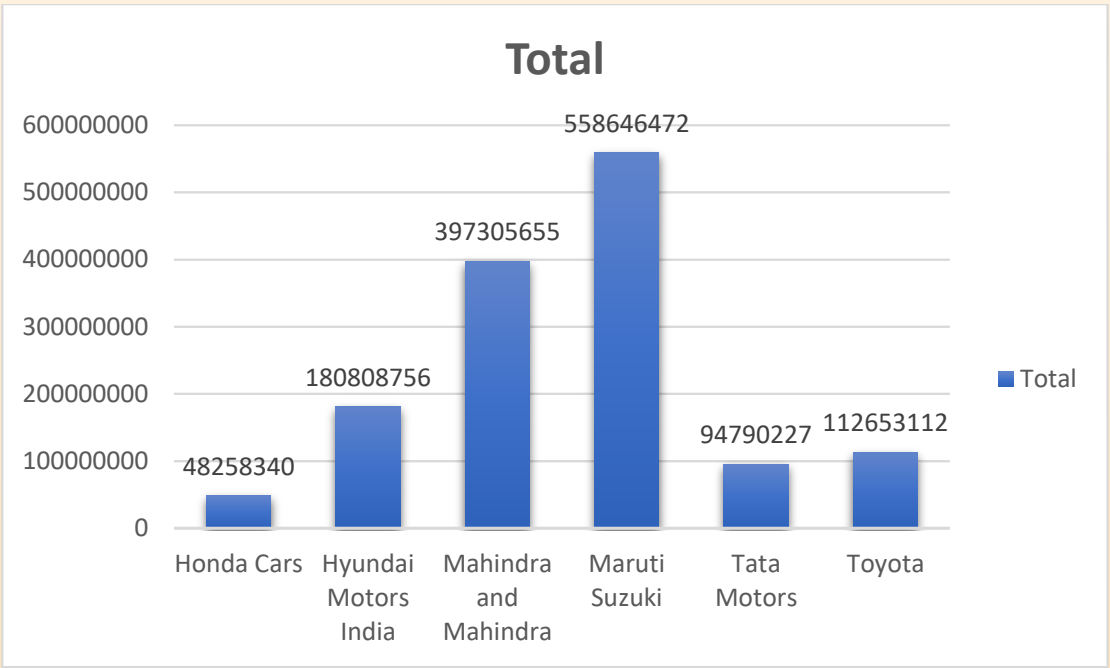
Same thing goes for Day part as well, brands have spent less in Q4 than Q1.



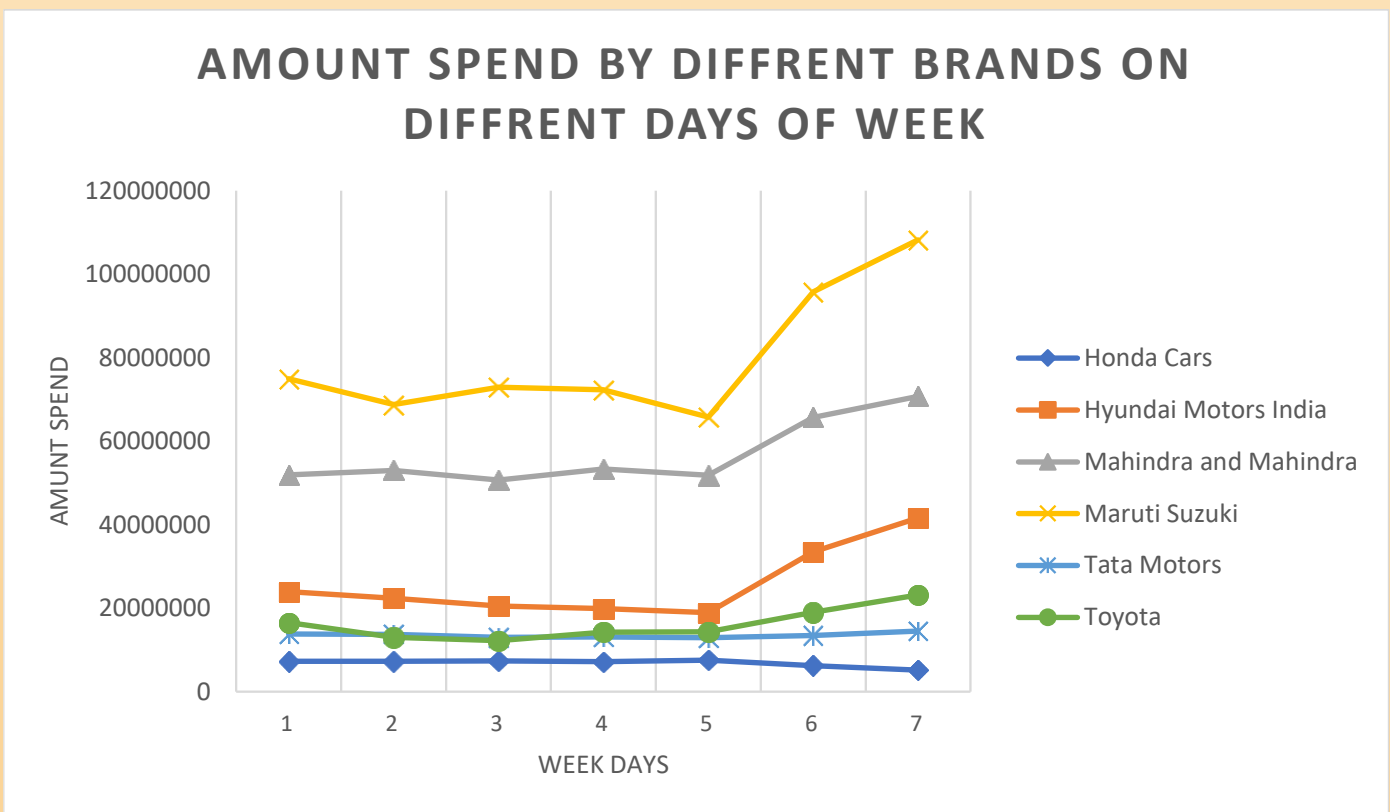
In conclusion, Brands spent more in Q1 of 2021 than Q4.

One reason might be because of 2nd wave of COVID 19 in India which led to many causalities as well as markets were highly affected.

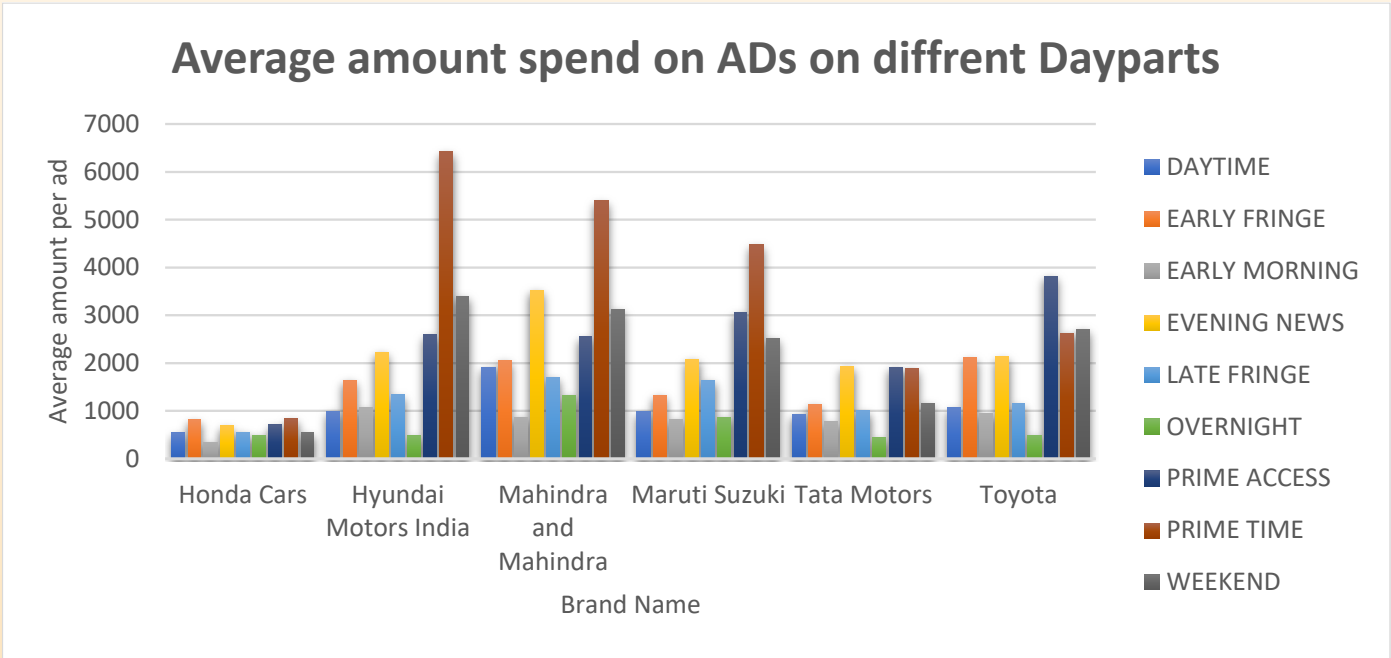
A. *Conduct a competitive analysis for the brands and define advertisement strategy of different brands and how it differs across the brands.*



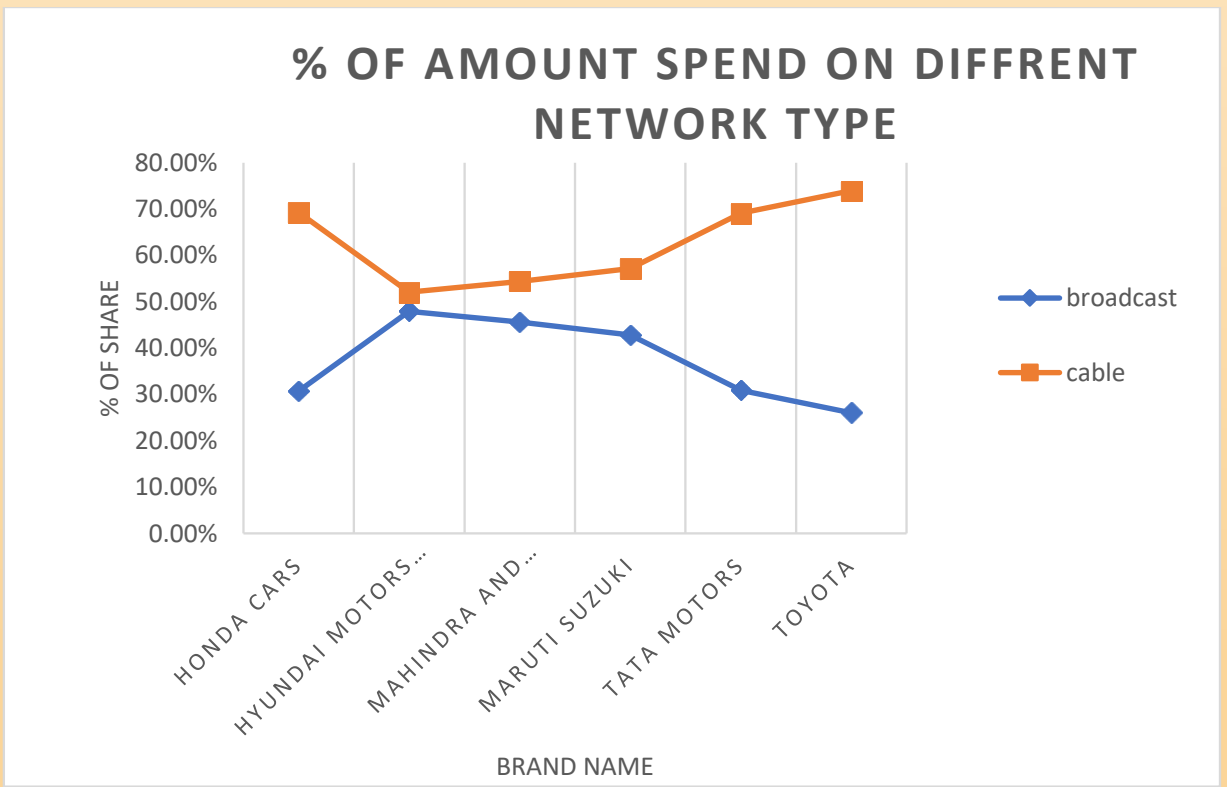
Maruti Suzuki has spent the most on ads, a total of 558,646,472.



The most amount of ads are shown on weekends by most of the brands.

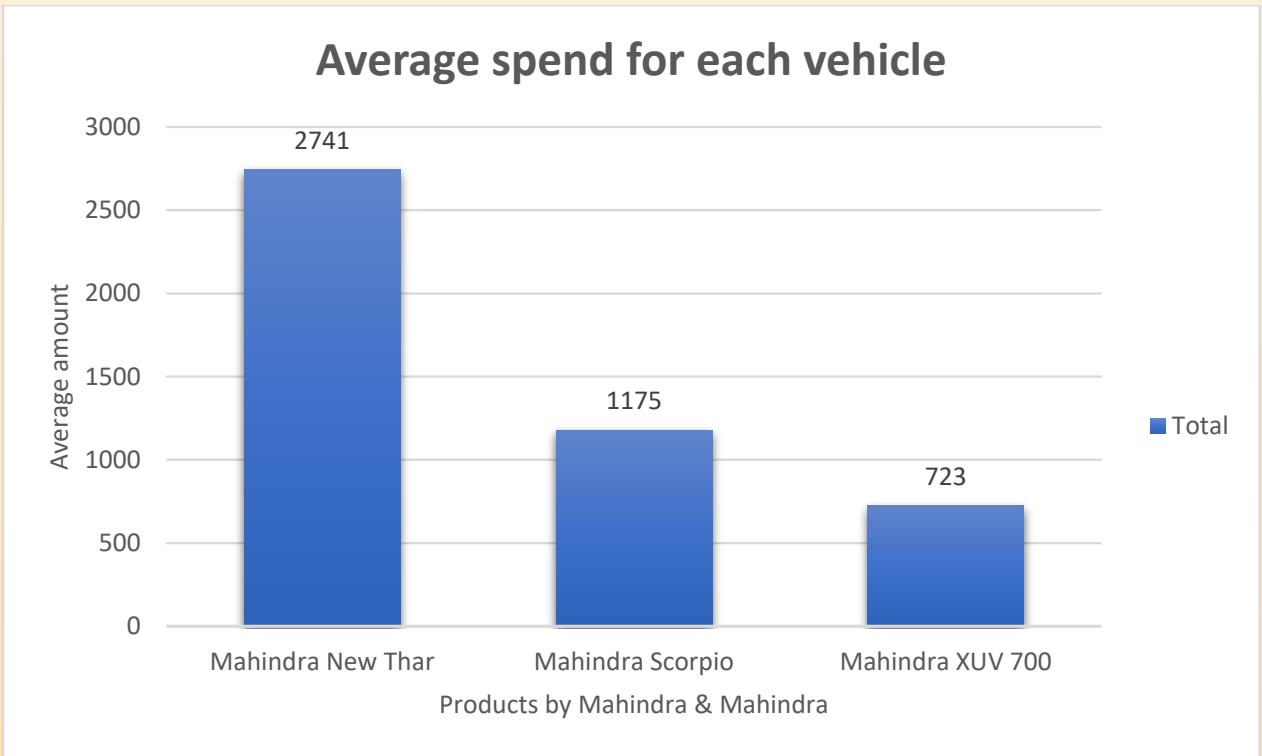


Day Parts such as -: Evening News, Prime Access, Prime Time and Weekend have highest average amount of ads.

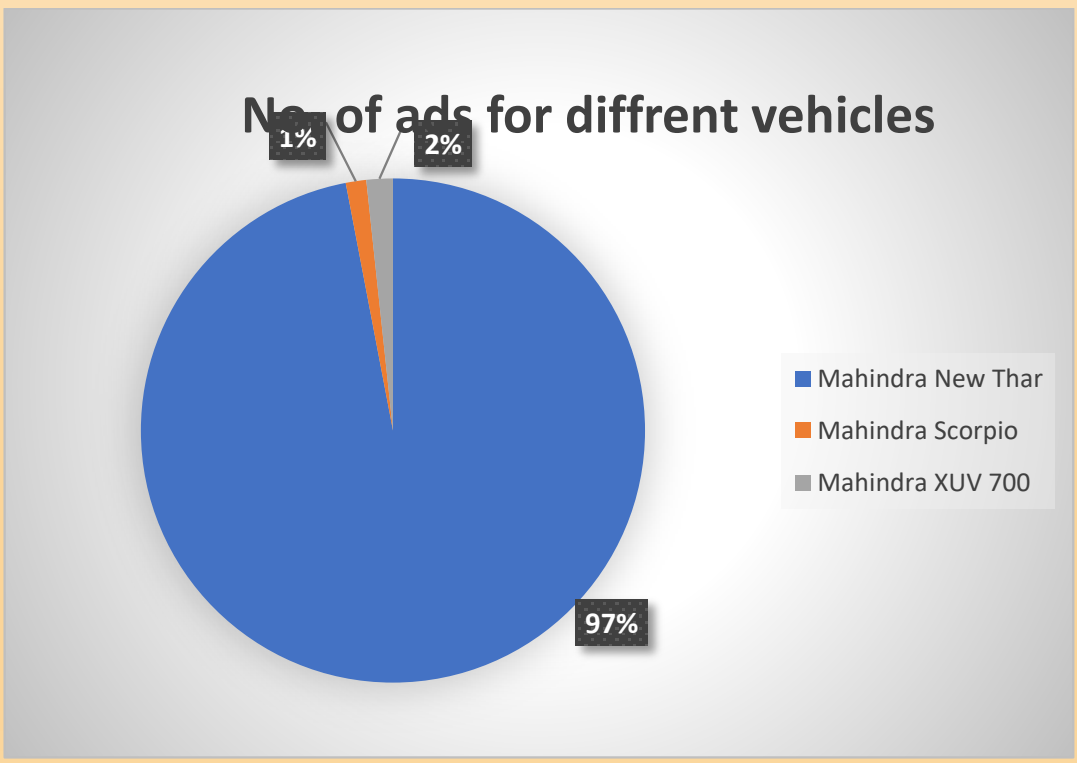


Amount spent on cable network is comparatively higher than broadcast by every brand.

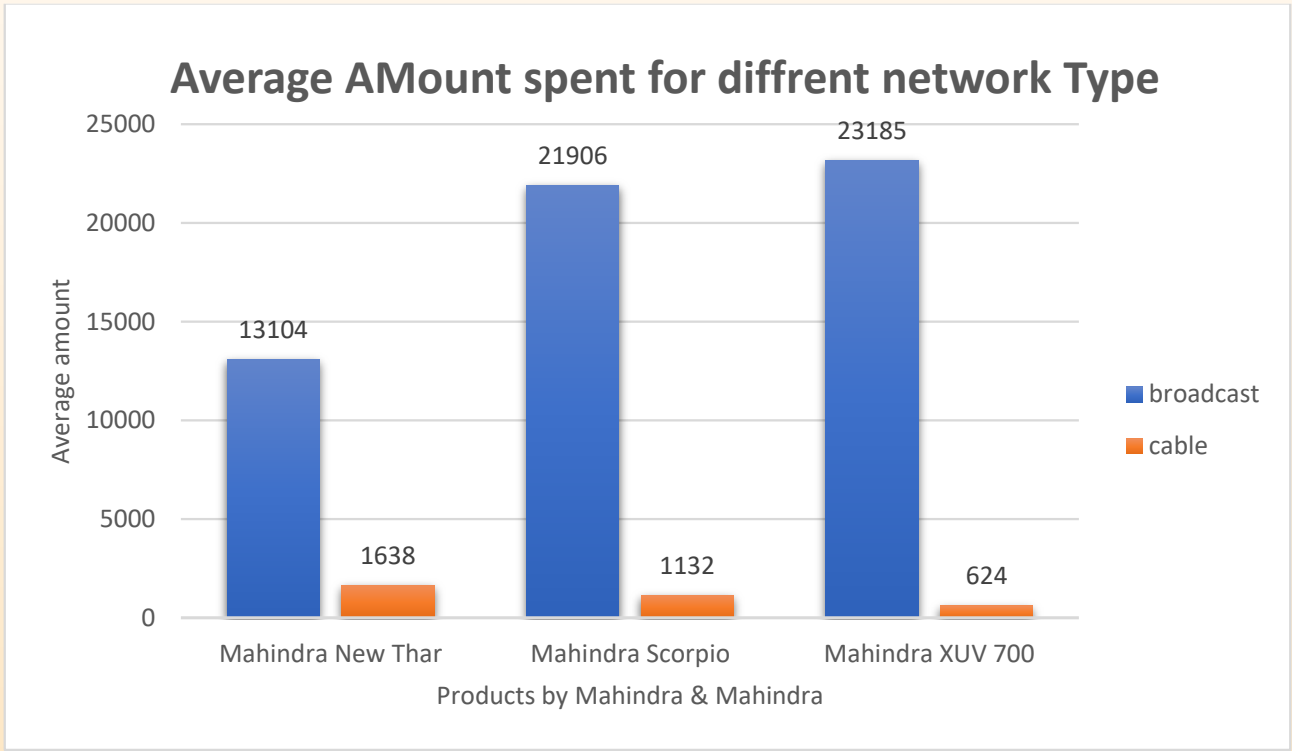
A. *Mahindra and Mahindra wants to run a digital ad campaign to complement its existing TV ads in Q1 of 2022. Based on the data from 2021, suggest a media plan to the CMO of Mahindra and Mahindra. Which audience should they target?*
**Assume XYZ Ads has the ad viewership data and TV viewership for the people in India.*
P.S. Brownie points for any additional actionable insights you can draw from the dataset.



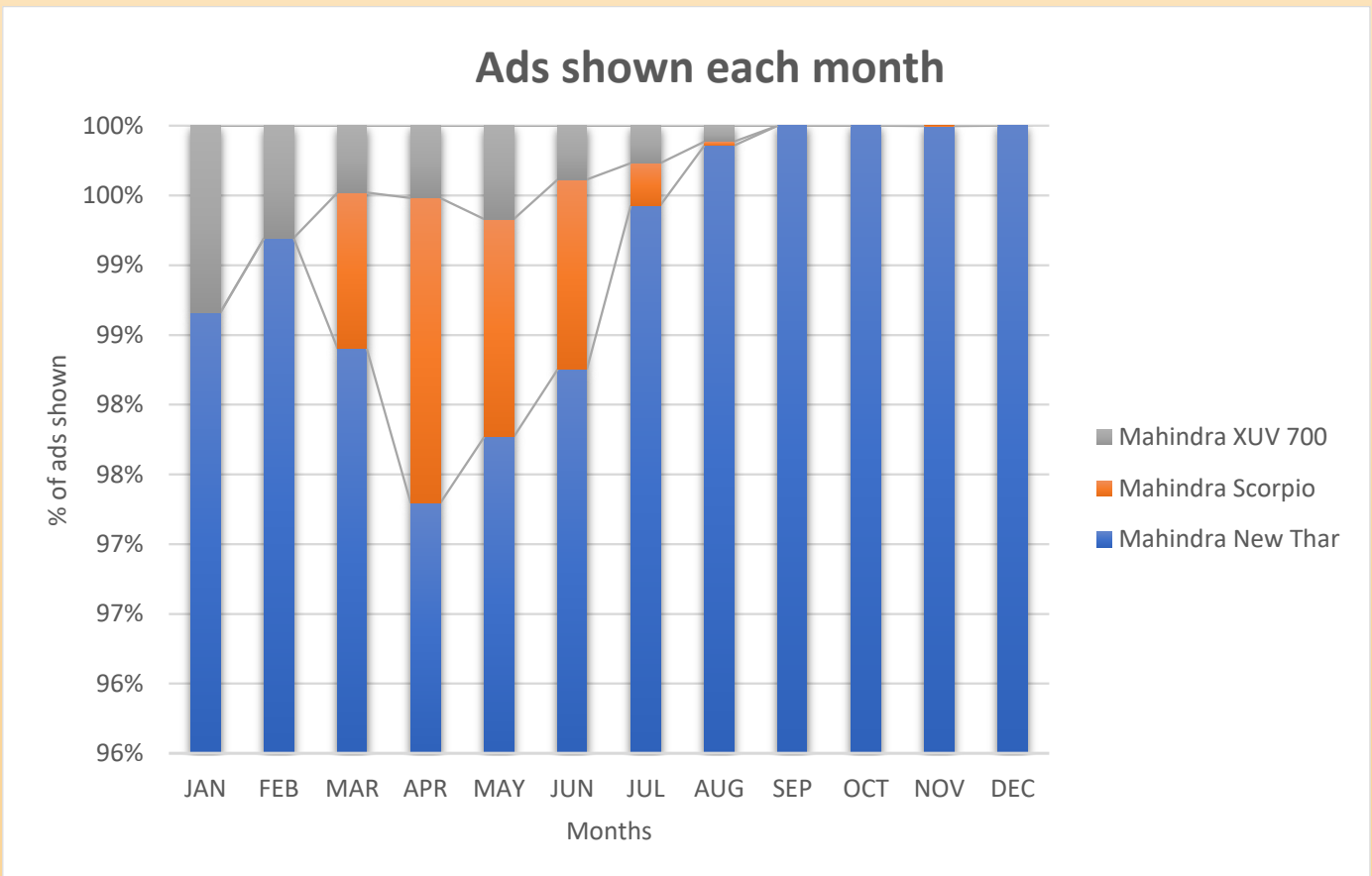
Mahindra & Mahindra spends highest on new vehicles which is why the amount spent on Mahindra Thar is the Highest.



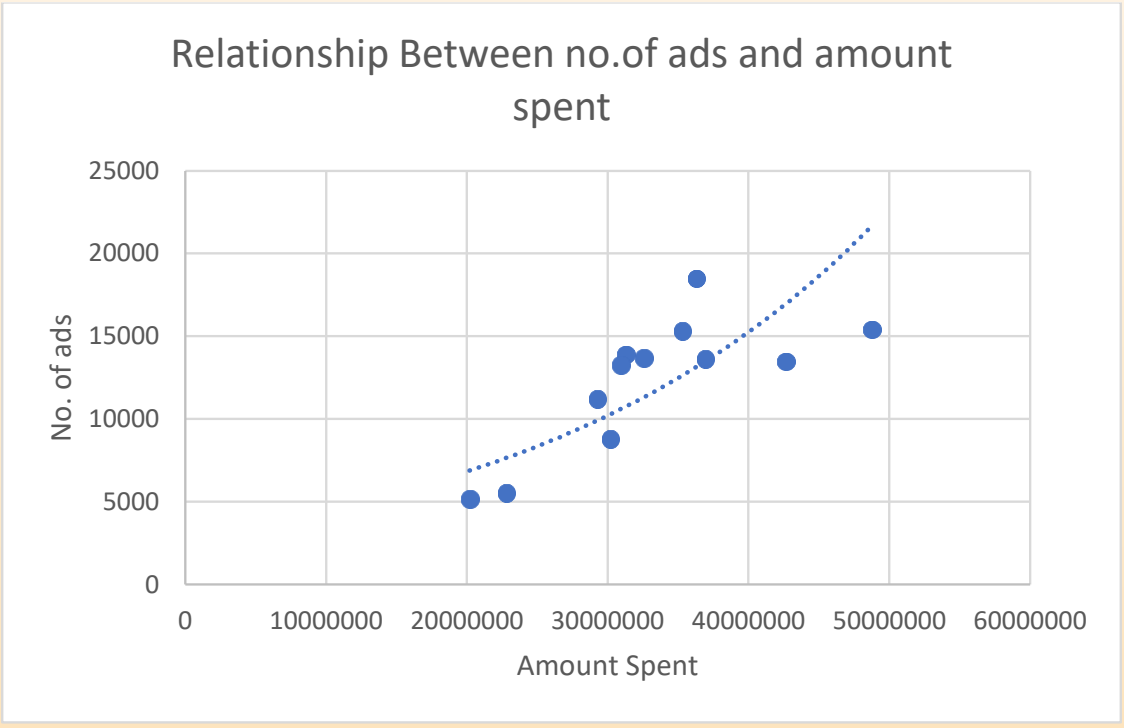
Out of total no. of ads, 97% of ads are of Mahindra Thar, 2% of Mahindra XUV 700 and only 1% of Mahindra Scorpio



The amount spent on broadcast network is way higher than cable. Whereas other brands don't have this much gap. This means that Mahindra does NOT believe in traditional way and invests in new technologies.



There's a fall in the % of ads of Scorpio and XUV 700, there reason here might again be second wave of COVID 19. Mahindra stopped spending money on old vehicle but kept spending on THAR.



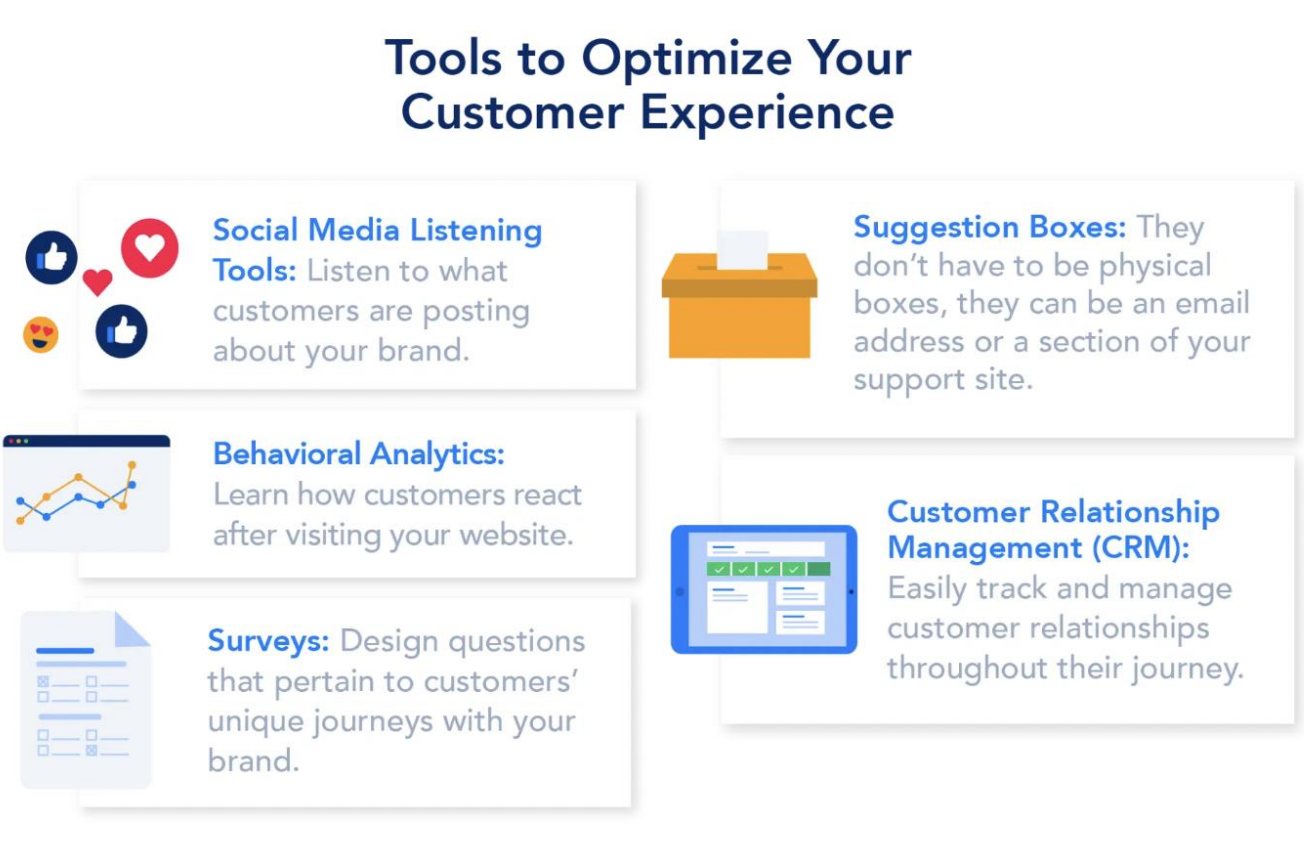
There is a relationship between the number of ads and the amount spent. I have used an exponential trendline to analyse this data.

My recommendation for 2022 ads campaign:-

Based on the analysis of the TV advertising dataset, it is recommended that Mahindra focuses its advertising efforts in 2022 on the target audience that is most likely to be interested in the new Thar. Specifically, the data suggests that Mahindra should target outdoor enthusiasts and adventure seekers who value off-road capabilities and performance. Given the high level of spending on Thar advertising, it appears that Mahindra is already aware of this target audience and sees potential for growth in this segment. However, in order to fully capitalize on this opportunity, it may be necessary to tailor advertising messages and channels to effectively reach and engage this target audience.

Module - 8

ABC Call Volume Trends Analysis



Project Description :

For you final project we are providing you with a dataset of a Customer Experience (CX) Inbound calling team for 23 days. Data includes Agent_Name, Agent_ID, Queue_Time [duration for which customer have to wait before they get connected to an agent], Time [time at which call was made by customer in a day], Time_Bucket [for easiness we have also provided you with the time bucket], Duration [duration for which a customer and executives are on call, Call_Seconds [for simplicity we have also converted those time into seconds], call status (Abandon, answered, transferred).

A customer experience (CX) team consists of professionals who analyze customer feedback and data, and share insights with the rest of the organization. Typically, these teams fulfil various roles and responsibilities such as: Customer experience programs (CX programs), Digital customer experience, Design and processes, Internal communications, Voice of the customer (VoC), User experiences, Customer experience management, Journey mapping, Nurturing customer interactions, Customer success, Customer support, Handling customer data, Learning about the customer journey.

Approach :

- Start with a clear and concise summary of your insights
- Use clear and concise language
- Use visual aids
- Provide context
- Highlight key findings
- Consider the audience

Tech Stack Used :

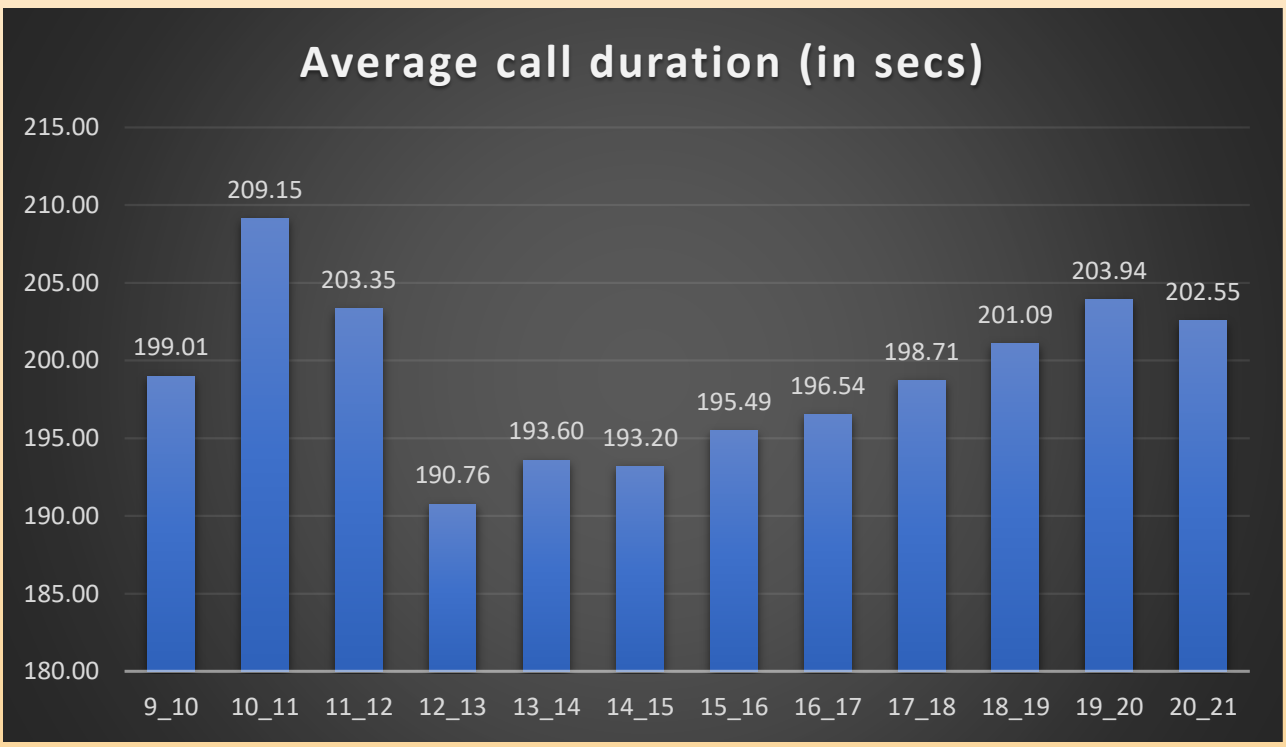
1. MS Excel

Link to excel File :

Click Here

A. Calculate the average call time duration for all incoming calls received by agents (in each Time_Bucket).

Time Bucket	Average call duration (in secs)
9_10	199.01
10_11	209.15
11_12	203.35
12_13	190.76
13_14	193.60
14_15	193.20
15_16	195.49
16_17	196.54
17_18	198.71
18_19	201.09
19_20	203.94
20_21	202.55



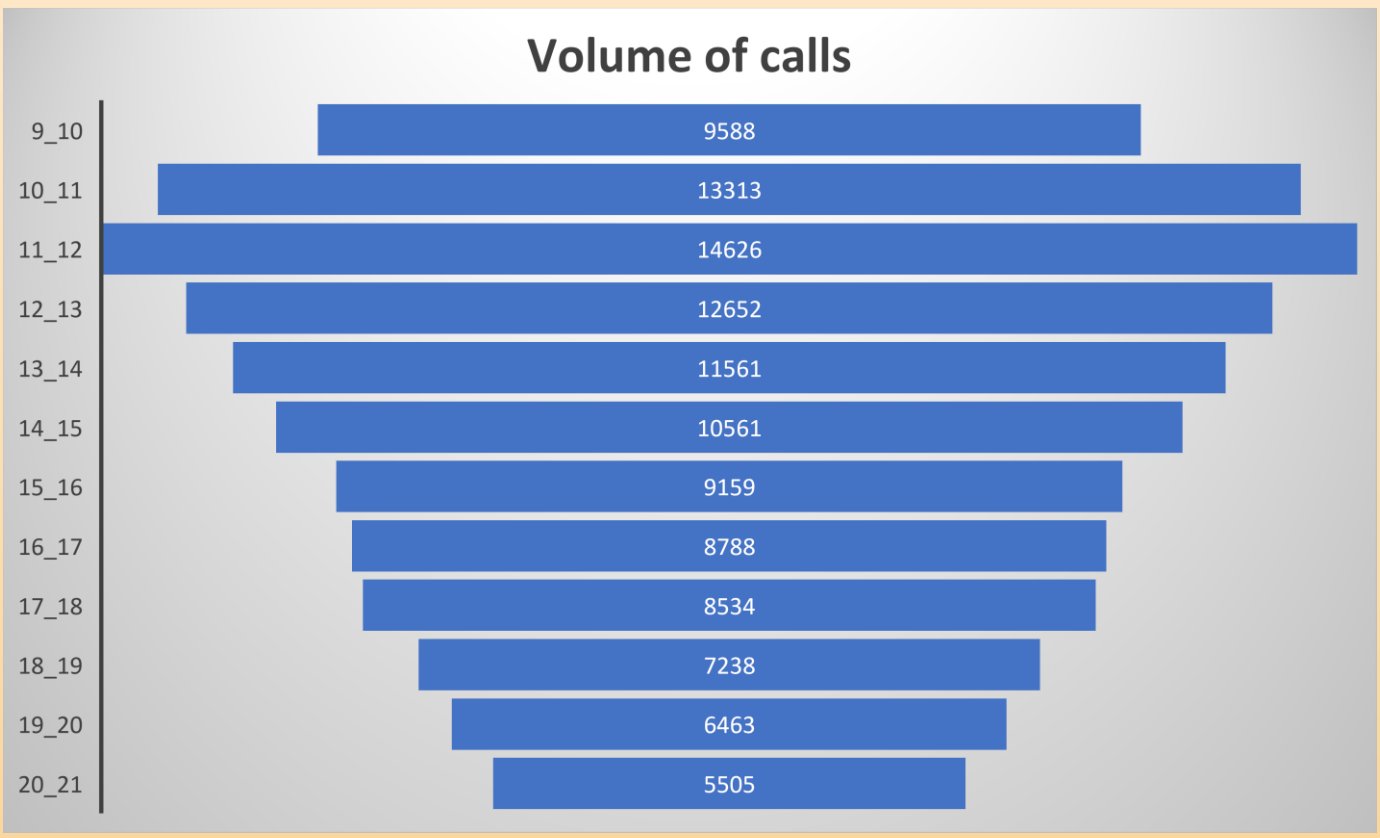
Insights:

- Time_Bucket is measured in the Rows and average of Call_Seconds is measured in the Values section. And we put Call_Status in the Filters section.
- The total average of call time duration which are answered by the agents is 198.6 seconds.
- The average call time duration for all incoming calls received by agents is the highest in between 10 am to 11 am and from 7 pm to 8 pm

The average call time duration for all incoming calls received by agents is the least in between 12 noon to 1 pm.

B. Show the total volume/ number of calls coming in via charts/ graphs [Number of calls v/s Time]. You can select time in a bucket form (i.e. 1-2, 2-3, …..)

Time Bucket	No. of calls
9_10	9588
10_11	13313
11_12	14626
12_13	12652
13_14	11561
14_15	10561
15_16	9159
16_17	8788
17_18	8534
18_19	7238
19_20	6463
20_21	5505



Insights:

- We plotted Time_Bucket in the rows and took Count of Customer_Phone_No and Count of Time in the Values section.
- We measured Count of Time as the percentage of Column Total.
- The customers call the most in between 11 am to 12 noon.
- The customers call the least in between 8 pm to 9 pm.

C. As you can see current abandon rate is approximately 30%. Propose a manpower plan required during each time bucket [between 9am to 9pm] to reduce the abandon rate to 10%. (i.e. You have to calculate minimum number of agents required in each time bucket so that at least 90 calls should be answered out of 100.)

Time Bucket	Number of calls	% of calls inbound	Required Agents
9_10	9588	8%	5
10_11	13313	11%	6
11_12	14626	12%	7
12_13	12652	11%	6
13_14	11561	10%	6
14_15	10561	9%	5
15_16	9159	8%	4
16_17	8788	7%	4
17_18	8534	7%	4
18_19	7238	6%	3
19_20	6463	5%	3
20_21	5505	5%	3
Grand Total	117988	100%	57

Date	abandon	answered	transfer	Grand Total
01-Jan	684	3883	77	4644
02-Jan	356	2935	60	3351
03-Jan	599	4079	111	4789
04-Jan	595	4404	114	5113
05-Jan	536	4140	114	4790
06-Jan	991	3875	85	4951
07-Jan	1319	3587	42	4948
08-Jan	1103	3519	50	4672
09-Jan	962	2628	62	3652
10-Jan	1212	3699	72	4983
11-Jan	856	3695	86	4637
12-Jan	1299	3297	47	4643
13-Jan	738	3326	59	4123
14-Jan	291	2832	32	3155
15-Jan	304	2730	24	3058
16-Jan	1191	3910	41	5142
17-Jan	16636	5706	5	22347
18-Jan	1738	4024	12	5774
19-Jan	974	3717	12	4703
20-Jan	833	3485	4	4322
21-Jan	566	3104	5	3675
22-Jan	239	3045	7	3291
23-Jan	381	2832	12	3225
Grand Total	34403	82452	1133	117988
Average	1496	3585	49	5130
% of calls	29%	70%	1%	

Assumptions:		
Agents working Hrs	9	Hrs
Break	1.5	Hrs
Agents working Hrs on Floor	7.5	Hrs
Agents Time spent on calls	4.5	Hrs
Avg Call Volume per day(9:00 AM to 9:00 PM)	5130	Nos
Average call duration (9:00 AM to 9:00 PM)	199	Seconds
Total call duration for 90% Calls	255	Hrs
Agents required per day	57	Nos

Insights

- Total agents working can be calculated by average calls on a single day divided by total time spend by one man in a single day.

  total agent = 187.96/5 = 37.59
- If agents are working for 5 hrs a day and 60% calls are getting answered. If we want 90% of the calls to get connected, we apply unitary method to find how many more employee we want.
- total agent = 90*37.59/60 = ~57 agents

1. First, we created pivot table. Date & Time is dragged down to Rows, Call Status to Columns, while taking count Call Duration in the Values section.
2. Then, we calculated the average of abandon, answered and transfer by using the average excel formula.
3. 29% of the calls are abandoned, 1% is transferred, while 70% of the calls are answered in the daytime.
4. Total agents required to answer the 90% of the calls per day is 56.
5. The minimum number of agents required for each time bucket is calculated by 56 * count of time (calculated in the 2nd question).

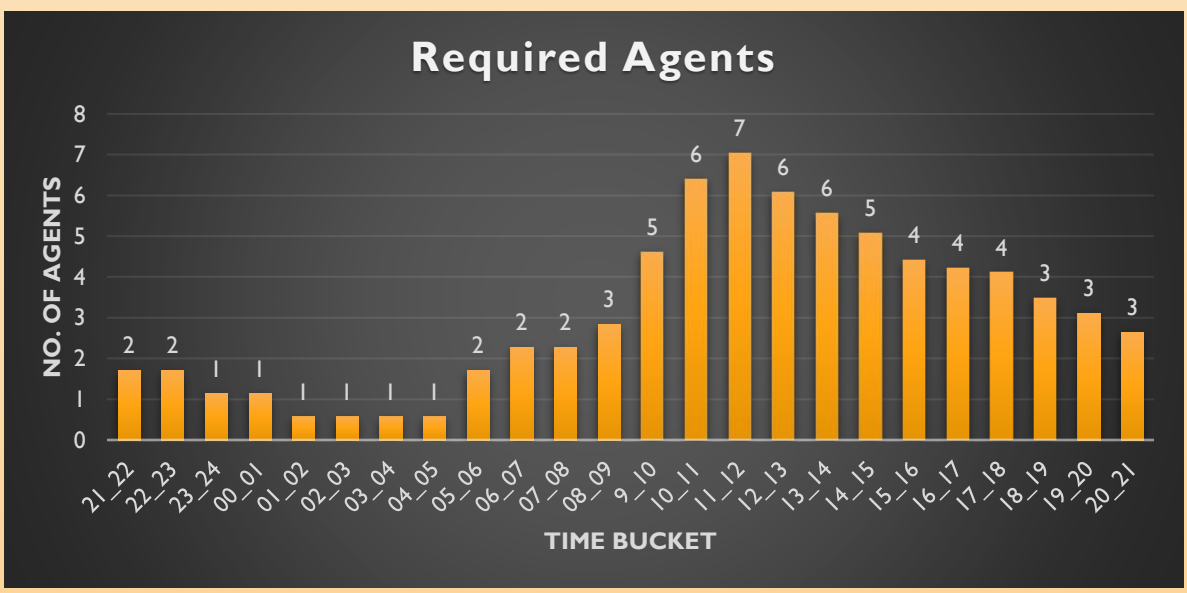
D. Let’s say customers also call this ABC insurance company in night but didn’t get answer as there are no agents to answer, this creates a bad customer experience for this Insurance company. Suppose every 100 calls that customer made during 9 Am to 9 Pm, customer also made 30 calls in night between interval [9 Pm to 9 Am] and distribution of those 30 calls are as follows:

Now propose a manpower plan required during each time bucket in a day. Maximum Abandon rate assumption would be same 10%.

Assumptions:		
Agents working Hrs	9	Hrs
Break	1.5	Hrs
Agents working Hrs on Floor	7.5	Hrs
Agents Time spent on calls	4.5	Hrs
Avg Call Volume per day(9:00 AM to 9:00 PM)	5130	Nos
Average call duration (9:00 AM to 9:00 PM)	199	Seconds
Total call duration for 90% Calls	255	Hrs
Agents required per day	57	Nos
If support is provided from 9:00 PM to 9:00 AM)		
Avg Call Volume (9:00 PM to 9:00 AM)	1539	Nos
Total Call duration (9:00 PM to 9:00AM)	77	Hours
Agents required (9:00 PM to 9:00AM)	17	Nos

Time_Bucket	Calls_inbound	% of calls inbound	Required agents
21_22	3	10%	2
22_23	3	10%	2
23_24	2	7%	1
00_01	2	7%	1
01_02	1	3%	1
02_03	1	3%	1
03_04	1	3%	1
04_05	1	3%	1
05_06	3	10%	2
06_07	4	13%	2
07_08	4	13%	2
08_09	5	17%	3
Total	30	100%	17

Time_Bucket	Required Agents
21_22	2
22_23	2
23_24	1
00_01	1
01_02	1
02_03	1
03_04	1
04_05	1
05_06	2
06_07	2
07_08	2
08_09	3
9_10	5
10_11	6
11_12	7
12_13	6
13_14	6
14_15	5
15_16	4
16_17	4
17_18	4
18_19	3
19_20	3
20_21	3



1. We first calculated the Time Distribution by dividing each calls distribution by total calls i.e., 30.
2. The number of agents required for each time bucket is calculated by 15 * Time Distribution
3. 15 is calculated above by dividing the additional hours required to answer the night calls by 5 (actual working hours of agents).
4. Also, while calculating, the round figure is taken into consideration as there cannot be 1.5 men working.

Insights:

1. The customers call the least in the evening. So, the company can reduce the number of agents at that time for answering the calls.
2. The company can hire 15 customer support agents for the night shift work.
3. The company can shift some of the day workers for the night shift.
4. The employees who are working 9 am to 9 pm. The manager can change some of the workers shift from 5 am to 2 pm and some workers from 2 pm to 11 pm to get the most calls answered.
5. The company can make the employers divide into 3 parts too, so that the agents are always available 24/7.
6. We found there were few outliers in the data. And if we have removed that outliers, then the answers would have been different.

Results:

1. I learned how an analyst can make an impact in customer service department.
2. I learned how a company deals with the customers to give them the most satisfaction.

THANK YOU