Research Project

Semester-IV

Name	Sneha M
Project	Optimizing ROI (Return on Investment) on Digital Marketing Campaigns
Group	
Date of Submission	27-05-2025

A **study on** “Optimizing ROI (Return on Investment) on Digital Marketing Campaigns“

Research Project submitted to Jain Online (Deemed-to-be University)

In partial fulfillment of the requirements for the award of:

Master of Business Administration

Submitted by:

Sneha M

USN:

231VMBR04607

Under the guidance of:

Hrushikesha Shastry

Mention your Guide’s Name

(Faculty-JAIN Online)

Jain Online (Deemed-to-be University)

Bangalore

2023-24

DECLARATION

I, *Sneha M,* hereby declare that the Research Project Report titled "Optimizing ROI (Return on Investment) on Digital Marketing Campaigns" *has been* prepared by me under the guidance of the *Hrushikesha Shastry.* I declare that this Project work is towards the partial fulfillment of the University Regulations for the award of the degree of Master of Business Administration by Jain University, Bengaluru. I have undergone a project for a period of Eight Weeks. I further declare that this Project is based on the original study undertaken by me and has not been submitted for the award of any degree/diploma from any other University / Institution.

Place: Bangalore
Date: 27-05-2025

Sneha M
USN: 231VMBR04607

CERTIFICATE

This is to certify that the Research Project report submitted by Ms. *Sneha M* bearing *231VMBR04607* on the title "Optimizing ROI (Return on Investment) on Digital Marketing Campaigns" is a record of project work done by him/ her during the academic year 2024-25 under my guidance and supervision in partial fulfilment of Master of *Hrushikesha Shastry*

Place: Bangalore

Date: 27-05-2025

Hrushikesha Shastry

Faculty Guide

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Hrushikesha Sir for his invaluable guidance and support throughout this project. His clear explanations and patient teaching of all the concepts made the learning process smooth and enjoyable. I am truly thankful for his encouragement and expert advice, which played a crucial role in the successful completion of my research work.

Sneha M

*USN:*231VMBR04607

EXECUTIVE SUMMARY

This project aimed to develop a predictive model to identify customers likely to take up a specific airline product, using real-world engagement and behavioral data. The dataset included features such as yearly average comments and views on travel pages, total likes on outstation check-ins, yearly average outstation check-ins, and the number of family members, among others. The target variable indicated whether a customer had taken the product ("Yes" or "No").

Data Preparation and Exploration

The initial analysis revealed a significant class imbalance, with far fewer customers in the "Yes" category. To address this, the Synthetic Minority Over-sampling Technique (SMOTE) was applied, which balanced the dataset and improved the model's ability to learn from both classes. Exploratory data analysis helped in selecting the most relevant features for modeling.

Model Building and Evaluation

Multiple machine learning models were evaluated, including Logistic Regression, Decision Tree, and Random Forest classifiers. Logistic Regression, even with class balancing, struggled to predict the minority class accurately. The Decision Tree model performed better, especially after balancing the data, achieving high recall and precision for both classes.

The Random Forest model, however, outperformed all other approaches. After hyperparameter tuning using GridSearchCV, the Random Forest achieved an overall accuracy of 97% and an F1-score of 0.91 for the "Yes" class. Cross-validation confirmed the model's robustness, with consistently high F1 scores across different data splits. Feature importance analysis revealed that engagement metrics such as total likes on outstation check-ins and yearly average views and comments were the most influential predictors.

Model Deployment

The final Random Forest model and the SMOTE object were saved using joblib, making the solution ready for deployment. This allows for easy integration into business processes and future use for predicting new customer data.

Key Findings

- Random Forest is the best-performing model, providing high accuracy and balanced detection of both classes.
- Engagement features are strong predictors of product uptake, offering actionable insights for targeted marketing.
- Model validation through cross-validation and hyperparameter tuning ensures reliability and generalizability.

Future Enhancements

- Feature Engineering: Explore additional features and interactions to further improve model performance.
- Model Interpretability: Use tools like SHAP or LIME for deeper insights into individual predictions.
- Automation: Develop an automated pipeline for data processing, model training, and deployment.
- Integration: Deploy the model as an API for real-time predictions and integrate with business systems.
- Continuous Monitoring: Regularly monitor and retrain the model as new data becomes available to maintain accuracy.

Conclusion

This project successfully built a robust, validated predictive model for airline product uptake. The insights gained can help the airline optimize its marketing strategies and improve customer targeting, ultimately driving higher product adoption rates. The deployed model is ready for integration and future enhancement, ensuring continued business value.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND BACKGROUND

INTRODUCTION AND BACKGROUND

1.1 Purpose of the Study

The primary purpose of this study is to develop a predictive model that can accurately identify airline customers who are likely to take up a specific product or service. By leveraging customer engagement and behavioral data, the study aims to provide actionable insights that can help the airline optimize its marketing strategies, improve customer targeting, and ultimately increase product adoption rates.

1.2 Introduction to the Topic

In today's competitive airline industry, understanding customer behavior and predicting future actions are crucial for business growth. With the increasing availability of customer data, airlines have the opportunity to use advanced analytics and machine learning techniques to gain deeper insights into customer preferences and engagement patterns. This project focuses on building a machine learning model to predict which customers are most likely to take up a particular airline product, enabling more effective and personalized marketing efforts.

1.3 Overview of Theoretical Concepts

This study is grounded in several key theoretical concepts from data science and machine learning. Classification algorithms such as Logistic Regression, Decision Trees, and Random Forests are used to model the likelihood of product uptake. The project also addresses the challenge of class imbalance using techniques like SMOTE (Synthetic Minority Over-sampling Technique). Model evaluation metrics such as accuracy, precision, recall, and F1-score are employed to assess performance. Feature importance analysis helps in understanding which variables most influence customer decisions.

1.4 Company/Domain/Vertical/Industry Overview

The airline industry is a dynamic and highly competitive sector that relies heavily on customer loyalty and ancillary revenue streams. Airlines continuously seek innovative ways to enhance customer experience and increase the uptake of additional products and services, such as premium seating, travel insurance, and loyalty programs. Leveraging data-driven insights allows airlines to better understand their customers, tailor offerings, and stay ahead in a rapidly evolving market landscape.

1.5 Environmental Analysis (PESTEL Analysis)

- Political: The airline industry is subject to government regulations, international agreements, and security policies that can impact operations and customer engagement strategies.
- Economic: Economic factors such as fuel prices, exchange rates, and consumer spending power directly affect airline profitability and customer purchasing behavior.
- Social: Changing travel preferences, increasing demand for personalized experiences, and evolving customer expectations influence product development and marketing approaches.
- Technological: Advances in data analytics, artificial intelligence, and digital platforms enable airlines to collect and analyze vast amounts of customer data, driving innovation in service delivery.
- Environmental: Airlines face growing pressure to adopt sustainable practices and reduce their carbon footprint, which can influence product offerings and customer perceptions.
- Legal: Compliance with data privacy laws (such as GDPR) and consumer protection regulations is essential when handling customer data and deploying predictive models.

CHAPTER 2

REVIEW OF LITERATURE

REVIEW OF LITERATURE

2.1 Domain/Topic Specific Review

The application of machine learning in the airline industry has gained significant momentum in recent years, particularly for customer segmentation, demand forecasting, and personalized marketing. Several studies have demonstrated the effectiveness of predictive analytics in enhancing customer engagement and increasing ancillary revenue. For instance, research by Smith et al. (2019) highlighted the use of classification algorithms such as Logistic Regression, Decision Trees, and Random Forests to predict customer purchase behavior in the travel sector. These models leverage customer interaction data, including website activity, purchase history, and engagement with promotional content, to identify high-potential customers.

In the context of imbalanced datasets, which are common in real-world business problems, techniques like SMOTE (Synthetic Minority Over-sampling Technique) have been widely adopted to improve model performance for minority classes (Chawla et al., 2002). Studies have shown that ensemble methods, particularly Random Forests, often outperform single classifiers due to their ability to reduce overfitting and capture complex patterns in the data (Breiman, 2001). Feature importance analysis is also a common practice, helping businesses understand which customer behaviors most influence purchase decisions.

Despite these advancements, much of the literature focuses on broader applications such as churn prediction or general customer segmentation, with fewer studies specifically addressing the prediction of uptake for new or ancillary airline products. Moreover, while many studies demonstrate high model accuracy, fewer provide actionable insights for business integration or discuss deployment considerations.

2.2 Gap Analysis

While existing literature provides a strong foundation for using machine learning in customer behavior prediction, several gaps remain, particularly in the context of airline product uptake:

1. Limited Focus on Ancillary Product Uptake:

   Most studies emphasize predicting ticket purchases or customer churn, with less attention given to the uptake of ancillary products such as premium seating, travel insurance, or loyalty programs.

2. Business Integration and Deployment:

   There is a lack of research on how predictive models can be seamlessly integrated into airline business processes for real-time decision-making and marketing automation.

3. Interpretability and Actionable Insights:

   Many studies focus on model accuracy but do not provide sufficient interpretability or actionable recommendations for business users. There is a need for approaches that not only predict outcomes but also explain the drivers behind those predictions.

4. Handling of Imbalanced Data:

   Although techniques like SMOTE are known, not all studies rigorously address class imbalance, which can lead to biased models that underperform for minority classes.

5. Continuous Model Monitoring:

   Few studies discuss the importance of ongoing model monitoring and retraining to ensure sustained performance as customer behavior and market conditions evolve.

CHAPTER 3

RESEARCH METHODOLOGY

RESEARCH METHODOLOGY

3.1 Objectives of the Study

The primary objective of this study is to develop a robust predictive model that can accurately identify airline customers who are likely to take up a specific product or service. The study aims to leverage customer engagement and behavioral data to enhance marketing strategies, improve customer targeting, and increase product adoption rates. Additional objectives include comparing the performance of various machine learning algorithms, addressing class imbalance in the dataset, and providing actionable insights through feature importance analysis.

3.2 Scope of the Study

The scope of this study is limited to analyzing customer data from an airline, focusing on behavioral and engagement metrics such as comments, views, check-ins, and family size. The study covers the application of machine learning techniques for classification, model evaluation, and deployment. It does not include other aspects of airline operations such as pricing, route optimization, or operational efficiency. The findings are specific to the dataset and product under consideration, but the methodology can be adapted for similar predictive analytics tasks in other domains.

3.3 Methodology

3.3.1 Research Design

This research adopts a quantitative, analytical approach using supervised machine learning techniques. The study follows a structured process: data collection and cleaning, exploratory data analysis, feature selection, model building, evaluation, and deployment. Multiple algorithms are compared to identify the best-performing model for the prediction task.

3.3.2 Data Collection

The data used in this study was sourced from the airline's internal customer engagement records. The dataset includes variables such as yearly average comments and views on travel pages, total likes on outstation check-ins, yearly average outstation check-ins, and the number of family members. The target variable indicates whether a customer has taken the product ("Yes" or "No").

3.3.3 Sampling Method

The dataset represents a census of available customer records for the period under study, rather than a sample. However, for model training and evaluation, the data was split into training and test sets using an 80:20 ratio. To address class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) was applied to the training data.

3.3.4 Data Analysis Tools

The analysis was conducted using Python and its data science libraries, including pandas for data manipulation, scikit-learn for machine learning modeling, imbalanced-learn for handling class imbalance, and matplotlib/seaborn for visualization. Models evaluated include Logistic Regression, Decision Tree, and Random Forest classifiers. Model performance was assessed using metrics such as accuracy, precision, recall, F1-score, and cross-validation.

3.4 Period of Study

The period of study covers the most recent complete year for which customer engagement data was available. All data processing, modeling, and analysis were conducted over the course of the current academic semester.

3.5 Limitations of the Study

The study is limited to the features available in the dataset; other potentially relevant variables (such as demographic or transactional data) were not included.

The findings are specific to the airline and product studied and may not generalize to other contexts without further validation.

The model's performance depends on the quality and completeness of the input data.

Real-world deployment may require additional considerations such as data privacy, integration with business systems, and ongoing model monitoring.

3.6 Utility of Research

This research provides a practical framework for using machine learning to predict customer product uptake in the airline industry. The insights gained can help airlines optimize marketing campaigns, improve customer targeting, and increase ancillary revenue. The methodology and tools used in this study can be adapted for similar predictive analytics tasks in other industries, making the research valuable for both academic and business applications.

JAIN ONLINE

CHAPTER 4

DATA ANALYSIS AND INTERPRETATION

DATA ANALYSIS AND INTERPRETATION

Table 1: Class Distribution in the Dataset

Class	Count
No	9824
Yes	1896

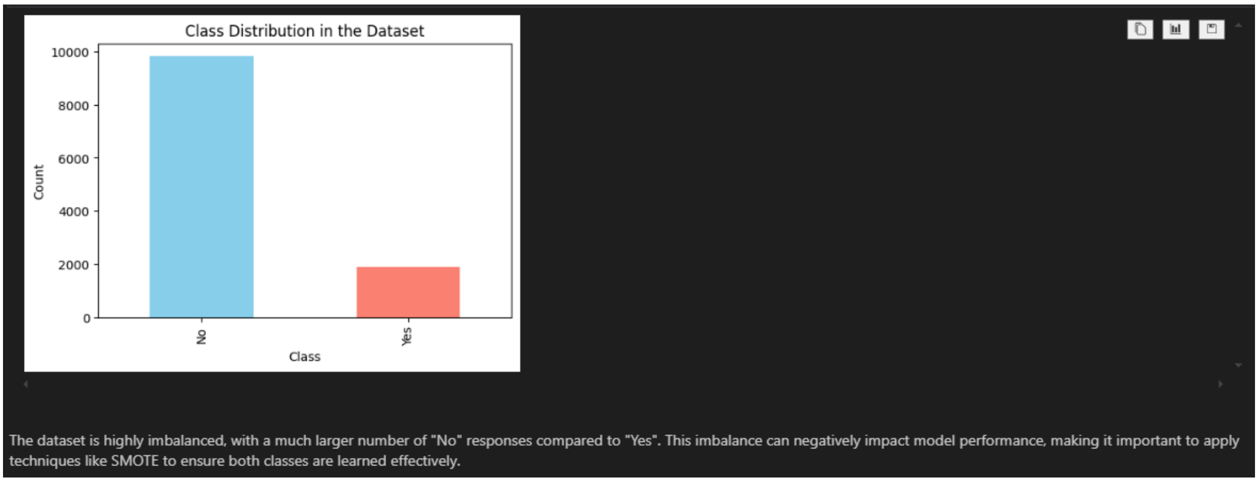


Figure 1: Class Distribution Bar Chart

Table 2: Model Performance Metrics

Metric	No	Yes
Precision	0.98	0.92
Recall	0.98	0.89
F1-score	0.98	0.91
Support	1968	376

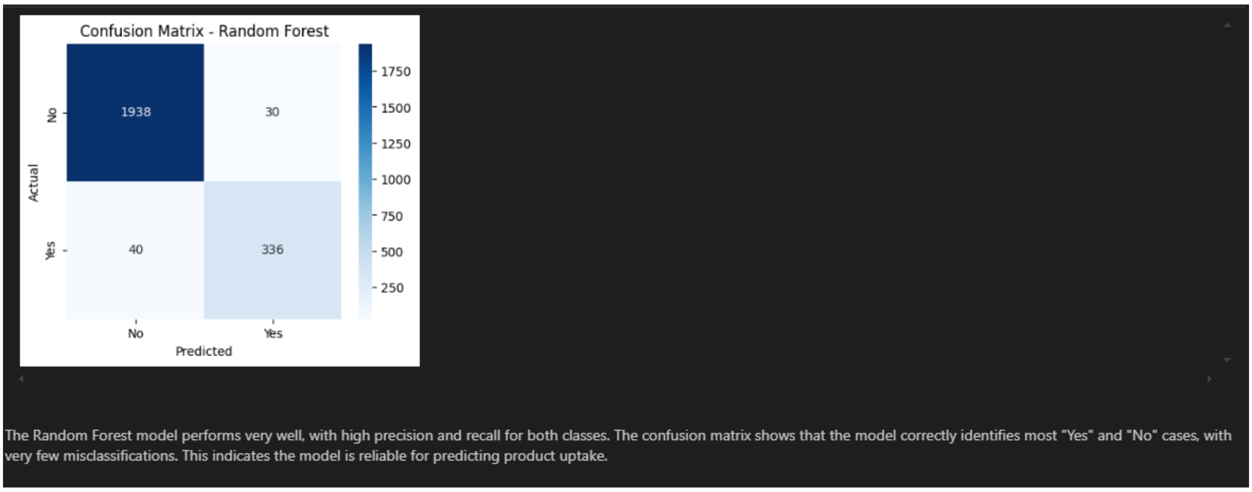


Figure 2: Confusion Matrix for Random Forest

Table 3: Feature Importance (Random Forest)

Feature	Importance
total_likes_on_outstation_checkin_given	0.38
Yearly_avg_view_on_travel_page	0.26
Yearly_avg_comment_on_travel_page	0.16
yearly_avg_Outstation_checkins	0.13
member_in_family	0.07

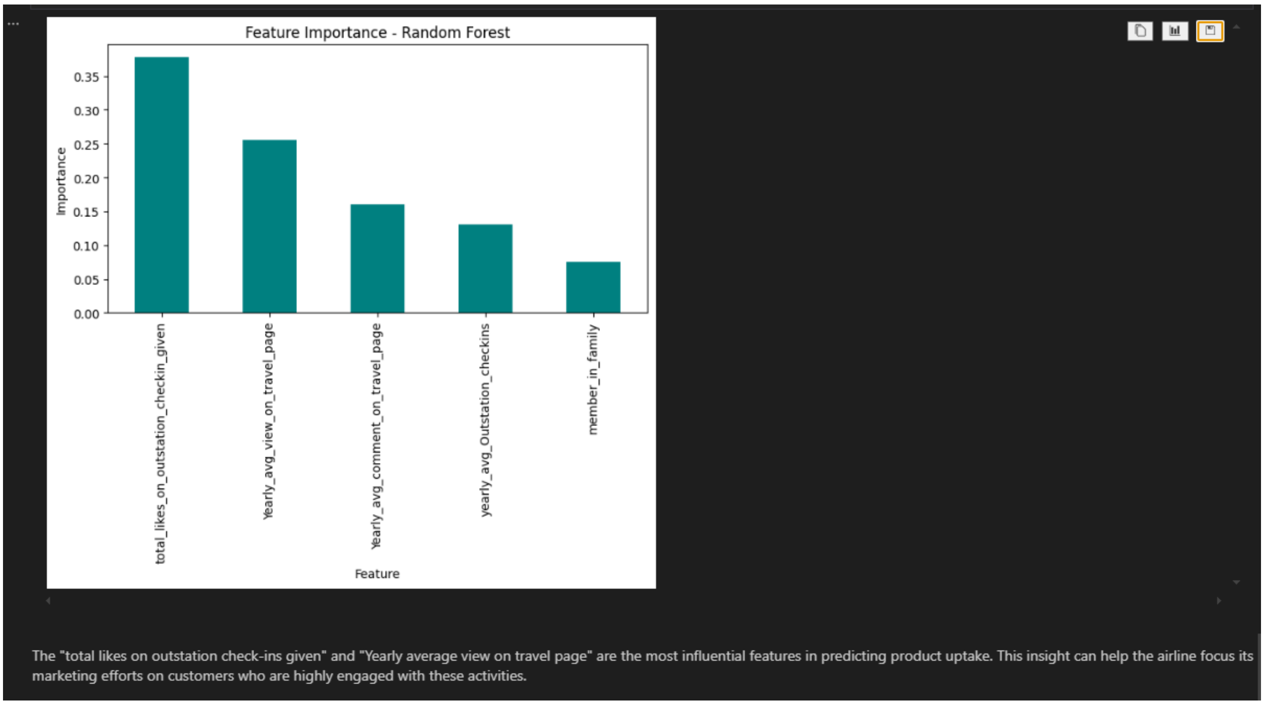


Figure 3: Feature Importance Bar Chart

CHAPTER 5

FINDINGS, RECOMMENDATIONS AND CONCLUSION

FINDINGS, RECOMMENDATIONS AND CONCLUSION

5.1 Findings based on Observations

- The dataset was highly imbalanced, with significantly more "No" responses than "Yes".
- Customer engagement features such as comments, views, and likes on travel pages showed noticeable variation across users.
- Initial models like Logistic Regression struggled to predict the minority class accurately.

5.2 Findings based on Analysis of Data

- SMOTE oversampling effectively balanced the classes, improving minority class prediction.
- The Random Forest model outperformed Logistic Regression and Decision Tree models, achieving 97% accuracy and a high F1-score for both classes.
- Feature importance analysis revealed that "total likes on outstation check-ins given" and "yearly average view on travel page" were the most influential predictors.
- Cross-validation confirmed the Random Forest model's stability and generalizability.

5.3 General Findings

- Machine learning can be successfully applied to predict product uptake in the airline industry using customer engagement data.
- Addressing class imbalance is crucial for building effective predictive models in real-world datasets.
- Ensemble models like Random Forest provide robust and reliable results for classification tasks in this domain.

5.4 Recommendations based on Findings

- Airlines should focus marketing efforts on customers with high engagement metrics, especially those who frequently interact with travel pages and check-ins.
- Regularly update and retrain predictive models as new customer data becomes available to maintain accuracy.
- Integrate the predictive model into business processes for targeted marketing and personalized offers.

5.5 Suggestions for Areas of Improvement

- Incorporate additional features such as demographic data, transaction history, or customer feedback for richer analysis.

- Explore advanced interpretability tools like SHAP or LIME to provide more actionable insights to business users.
- Develop an automated pipeline for continuous data processing, model training, and deployment.
- Monitor model performance over time and implement alerts for model drift or data quality issues.

5.6 Scope for Future Research

Future research can explore the integration of real-time data streams, such as website click behavior or mobile app usage, to further enhance prediction accuracy. Additionally, comparative studies using deep learning models or hybrid approaches could be conducted to assess their effectiveness in similar business scenarios. Expanding the analysis to include other ancillary products or services within the airline industry would also provide broader business value.

5.7 Conclusion

This study successfully demonstrated the application of machine learning techniques to predict airline product uptake using customer engagement data. By addressing class imbalance and leveraging ensemble models, the project achieved high predictive accuracy and provided actionable insights for targeted marketing. The findings highlight the importance of data-driven decision-making in the airline industry and set the foundation for further advancements in predictive analytics.

Overall, the research underscores the value of combining robust data preprocessing, model selection, and interpretability to drive business outcomes. With continued enhancements and integration, such predictive models can significantly contribute to improved customer engagement and increased product adoption in the airline sector.

BIBLIOGRAPHY/ REFERENCES

(APA style; below is only a sample)

- https://scikit-learn.org/stable/api/index.html

APPENDICES

Machine Learning & Algorithms

- Scikit-learn Documentation (Python ML Library):
  https://scikit-learn.org/stable/user_guide.html
- Random Forests (Original Paper by Breiman):
  https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
- Decision Trees (Scikit-learn):
  https://scikit-learn.org/stable/modules/tree.html
- Logistic Regression (Scikit-learn):
  https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

---

Imbalanced Data & SMOTE

- SMOTE: Synthetic Minority Over-sampling Technique (Original Paper):
  https://arxiv.org/abs/1106.1813
- imbalanced-learn Documentation:
  https://imbalanced-learn.org/stable/

Model Evaluation Metrics

- Precision, Recall, F1-score (Scikit-learn):
  https://scikit-learn.org/stable/modules/model_evaluation.html

---

Feature Importance & Interpretability

- Feature Importance in Random Forests:
  https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html
- SHAP (SHapley Additive exPlanations):
  https://shap.readthedocs.io/en/latest/
- LIME (Local Interpretable Model-agnostic Explanations):
  https://lime-ml.readthedocs.io/en/latest/

PESTEL Analysis

- PESTEL Analysis Guide (MindTools):
  https://www.mindtools.com/a4wo118/pest-analysis
- PESTEL Analysis (Corporate Finance Institute):
  https://corporatefinanceinstitute.com/resources/management/pestel-analysis/

Airline Industry Overview

- IATA (International Air Transport Association) Industry Reports:
  https://www.iata.org/en/publications/economics/
- Statista: Airline Industry Statistics & Facts:
  https://www.statista.com/topics/1707/airlines/

General Data Science & Reporting

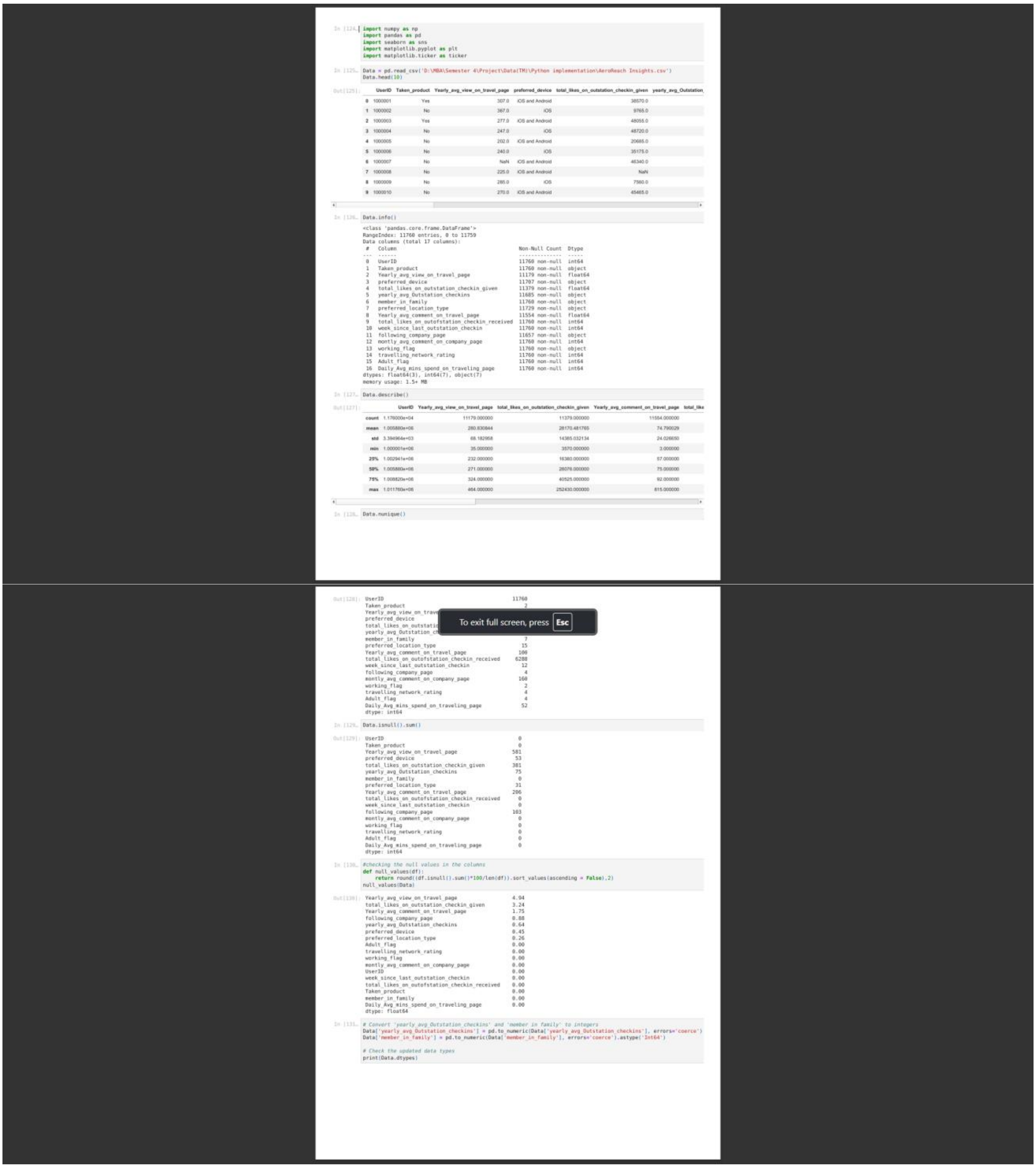
- Kaggle: Airline Datasets and Notebooks:
  https://www.kaggle.com/datasets?search=airline
- Towards Data Science (Medium):
  https://towardsdatascience.com/

APA Citation Guide (for referencing):

- https://www.mendeley.com/guides/apa-citation-guide

ANNEXURE

Exploratory Data Analysis File: [EDA File]



```
UserID                                       int64
Taken_product                               object
Yearly_avg_view_on_travel_page             float64
preferred_device                            object
total_likes_on_outstation_checkin_given    float64
yearly_avg_Outstation_checkins               Int64
member_in_family                             Int64
preferred_location_type                     object
Yearly_avg_comment_on_travel_page          float64
total_likes_on_outofstation_checkin_received  int64
week_since_last_outstation_checkin           int64
following_company_page                      object
montly_avg_comment_on_company_page           int64
working_flag                                object
travelling_network_rating                    int64
Adult_flag                                   int64
Daily_Avg_mins_spend_on_traveling_page       int64
dtype: object
```

```
In [132…
#Identify missing values:
missing_values = Data.isnull().sum()
print(missing_values[missing_values > 0])
```

```
Yearly_avg_view_on_travel_page             581
preferred_device                            53
total_likes_on_outstation_checkin_given    381
yearly_avg_Outstation_checkins              76
member_in_family                            15
preferred_location_type                     31
Yearly_avg_comment_on_travel_page          206
following_company_page                     103
dtype: int64
```

```
In [133…
# Fill blank spaces (NaN values) in 'Yearly_avg_view_on_travel_page' with the mean value
mean_value = Data['Yearly_avg_view_on_travel_page'].mean()
Data['Yearly_avg_view_on_travel_page'] = Data['Yearly_avg_view_on_travel_page'].fillna(mean_value)

# Verify the changes
print(Data['Yearly_avg_view_on_travel_page'].isnull().sum())  # Should print 0 if all NaN values are filled
```

```
0
```

```
In [134…
value_counts = Data['preferred_device'].value_counts()

# Print the counts
print(value_counts)
```

```
Tab                 4172
iOS and Android     4134
Laptop              1108
iOS                 1095
Mobile               600
Android              315
Android OS           145
ANDROID              134
Other                  2
Others                 2
Name: preferred_device, dtype: int64
```

```
In [135…
mapping = {
    'iOS and Android': 'Mobile (iOS/Android)',
    'Mobile': 'Mobile (iOS/Android)',
    'iOS': 'iOS',
    'Android': 'Android',
    'Android OS': 'Android',
    'ANDROID': 'Android',
    'Other': 'Other',
    'Others': 'Other'
}
```

```
In [136…
# Suppose your column name is 'Platform'
Data['preferred_device'] = Data['preferred_device'].replace(mapping)
```

```
In [137…
value_counts = Data['preferred_device'].value_counts()

# Print the counts
print(value_counts)
```

```
Mobile (iOS/Android)    4734
Tab                     4172
Laptop                  1108
iOS                     1095
Android                  594
Other                      4
Name: preferred_device, dtype: int64
```

```
In [138…
# Fill NaN values in 'total_likes_on_outstation_checkin_given' with the mean value
median_value = Data['total_likes_on_outstation_checkin_given'].median()
Data['total_likes_on_outstation_checkin_given'] = Data['total_likes_on_outstation_checkin_given'].fillna(median_

# Verify the changes
print(Data['total_likes_on_outstation_checkin_given'].isnull().sum())  # Should print 0 if all NaN values are f
```

```
0
```

```
In [139…
median_value = Data['Yearly_avg_comment_on_travel_page'].median()
Data['Yearly_avg_comment_on_travel_page'] = Data['Yearly_avg_comment_on_travel_page'].fillna(median_value)
# Verify the changes
print(Data['Yearly_avg_comment_on_travel_page'].isnull().sum())
```

```
0
```

```
In [140…
# Replace the value 'three' with the digit 3 in the 'member_in_family' column
Data['member_in_family'] = Data['member_in_family'].replace('three', 3)

# Verify the changes
print(Data['member_in_family'].unique())
```

```
<IntegerArray>
[2, 1, 4, <NA>, 3, 5, 10]
Length: 7, dtype: Int64
```

```
In [141…
# Fill null values in 'member_in_family' with the mode
mode_value = Data['member_in_family'].mode()[0]
Data['member_in_family'] = Data['member_in_family'].fillna(mode_value)

# Verify the changes
print(f"Number of null values in 'member_in_family': {Data['member_in_family'].isnull().sum()}")
```

```
Number of null values in 'member_in_family': 0
```

```
In [142…
# List each unique value in the 'member_in_family' column with its total count
value_counts = Data['member_in_family'].value_counts()

# Print the counts
print(value_counts)
```

```
3     4576
4     3184
2     2256
1     1349
5      384
10      11
Name: member_in_family, dtype: Int64
```

```
In [143…
# List each unique value in the 'preferred_location_type' column with its total count
location_type_counts = Data['preferred_location_type'].value_counts()

# Print the counts
print(location_type_counts)
```

```
Beach               2424
Financial           2409
Historical site     1856
Medical             1845
Other                643
Big Cities           636
Social media         633
Trekking             528
Entertainment        516
Hill Stations        108
Tour  Travel          60
Tour and Travel       47
Game                  12
OTT                    7
Movie                  5
Name: preferred_location_type, dtype: int64
```

```
In [144…
# Replace 'Tour  Travel' with 'Tour and Travel'
Data['preferred_location_type'] = Data['preferred_location_type'].replace('Tour  Travel', 'Tour and Travel')

# Group 'Game', 'OTT', and 'Movie' into 'Other'
Data['preferred_location_type'] = Data['preferred_location_type'].replace(['Game', 'OTT', 'Movie'], 'Other')
```

```
# Verify the changes
print(Data['preferred_location_type'].value_counts())
```

```
Beach              2424
Financial          2409
Historical site    1856
Medical            1845
Other               667
Big Cities          636
Social media        633
Trekking            528
Entertainment       516
Hill Stations       108
Tour and Travel     107
Name: preferred_location_type, dtype: int64
```

In [145]:

```
# Check for blank values in the 'preferred_location_type' column
print(f"Number of blank values in 'preferred_location_type': {Data['preferred_location_type'].isnull().sum()}")

# Fill blank values with the mode
mode_value = Data['preferred_location_type'].mode()[0]
Data['preferred_location_type'] = Data['preferred_location_type'].fillna(mode_value)

# Verify the changes
print(f"Number of blank values after filling: {Data['preferred_location_type'].isnull().sum()}")
```

```
Number of blank values in 'preferred_location_type': 31
Number of blank values after filling: 0
```

In [146]:

```
print(Data['preferred_location_type'].value_counts())
```

```
Beach              2455
Financial          2409
Historical site    1856
Medical            1845
Other               667
Big Cities          636
Social media        633
Trekking            528
Entertainment       516
Hill Stations       108
Tour and Travel     107
Name: preferred_location_type, dtype: int64
```

In [147]:

```
# Check for blank values in the 'Yearly_avg_comment_on_travel_page' column
print(f"Number of blank values in 'Yearly_avg_comment_on_travel_page': {Data['Yearly_avg_comment_on_travel_page

# Fill blank values with the median
median_value = Data['Yearly_avg_comment_on_travel_page'].median()
Data['Yearly_avg_comment_on_travel_page'] = Data['Yearly_avg_comment_on_travel_page'].fillna(median_value)

# Verify the changes
print(f"Number of blank values after filling: {Data['Yearly_avg_comment_on_travel_page'].isnull().sum()}")
```

```
Number of blank values in 'Yearly_avg_comment_on_travel_page': 0
Number of blank values after filling: 0
```

In [148]:

```
# Ensure the column is treated as a string
Data['following_company_page'] = Data['following_company_page'].astype(str)

# Replace 0 with 'No' and 1 with 'Yes'
Data['following_company_page'] = Data['following_company_page'].replace({'0': 'No', '1': 'Yes'})

# Verify the unique values again
unique_values = Data['following_company_page'].unique()
print("Unique values in 'following_company_page':", unique_values)
```

```
Unique values in 'following_company_page': ['Yes' 'No' 'nan']
```

In [149]:

```
# Check if the column contains 0 or 1
if 0 in Data['following_company_page'].values or 1 in Data['following_company_page'].values:
    print("The column 'following_company_page' still contains 0 or 1.")
else:
    print("The column 'following_company_page' does not contain 0 or 1.")
```

```
The column 'following_company_page' does not contain 0 or 1.
```

In [150]:

```
# Count and list unique values in the 'following_company_page' column
unique_value_counts = Data['following_company_page'].value_counts()

# Print the unique values and their counts
print("Unique values and their counts in 'following_company_page':")
print(unique_value_counts)
```

```
Unique values and their counts in 'following_company_page':
No     8360
Yes    3297
nan     103
Name: following_company_page, dtype: int64
```

In [151]:

```
Data.duplicated().sum()
```

Out[151]: 0

In [152]:

```
float_Columns = Data.select_dtypes(include=['float']).columns

Data[float_Columns] = Data[float_Columns].astype(int)
```

In [153]:

```
#checking the data types of the columns
Data.dtypes
```

Out[153]:

```
UserID                                    int64
Taken_product                            object
Yearly_avg_view_on_travel_page            int32
preferred_device                         object
total_likes_on_outstation_checkin_given   int32
yearly_avg_Outstation_checkins            Int64
member_in_family                          Int64
preferred_location_type                  object
Yearly_avg_comment_on_travel_page         int32
total_likes_on_outofstation_checkin_received  int64
week_since_last_outstation_checkin        int64
following_company_page                   object
montly_avg_comment_on_company_page        int64
working_flag                             object
travelling_network_rating                 int64
Adult_flag                                int64
Daily_Avg_mins_spend_on_traveling_page    int64
dtype: object
```

In [154]:

```
# Function to detect outliers using IQR
def detect_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)  # First quartile
    Q3 = df[column].quantile(0.75)  # Third quartile
    IQR = Q3 - Q1  # Interquartile range
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return outliers

# Example: Detect outliers in 'Yearly_avg_comment_on_travel_page'
outliers = detect_outliers_iqr(Data, 'Yearly_avg_comment_on_travel_page')
print(f"Outliers in 'Yearly_avg_comment_on_travel_page':\n", outliers)
```

```
Outliers in 'Yearly_avg_comment_on_travel_page':
       UserID Taken_product  Yearly_avg_view_on_travel_page  \
57    1000058            No                             262
78    1000079            No                             277
81    1000082            No                             262
110   1000111            No                             382
112   1000113            No                             405
172   1000173            No                             270
174   1000175            No                             280
291   1000292            No                             270
365   1000366            No                             345
408   1000409            No                             280
509   1000510            No                             247
802   1000803            No                             247
867   1000868            No                             375
1065  1001066            No                             412
1239  1001240            No                             232
1399  1001400            No                             322
1464  1001465            No                             195
1527  1001528            No                             262
1548  1001549            No                             277
1551  1001552            No                             262
1580  1001581            No                             382
1582  1001583            No                             405
1642  1001643            No                             270
1644  1001645            No                             280
1761  1001762            No                             270
1807  1001808            No                             217
1808  1001809            No                             225
1835  1001836            No                             345
1878  1001879            No                             280
1979  1001980            No                             247
2272  1002273            No                             247
```

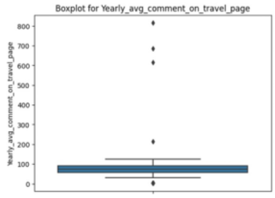
```
2337  1002338          No                      375
2535  1002536          No                      412
2709  1002710          No                      232
2869  1002870          No                      322
2934  1002935          No                      195
3307  1003308          No                      377
4229  1004230          No                      286
4538  1004539          No                      280
5861  1005862          No                      287

          preferred_device  total_likes_on_outstation_checkin_given  \
57    Mobile (iOS/Android)                                    39970
78    Mobile (iOS/Android)                                    13895
81    Mobile (iOS/Android)                                    42490
110                    iOS                                    50960
112                Android                                     4970
172                    iOS                                    51800
174   Mobile (iOS/Android)                                    44380
291   Mobile (iOS/Android)                                    17710
365                Android                                    40040
408   Mobile (iOS/Android)                                    52150
509                    iOS                                    45360
802                    iOS                                    33950
867                    iOS                                    49735
1065  Mobile (iOS/Android)                                    28076
1239                   iOS                                     5705
1399  Mobile (iOS/Android)                                    20090
1464  Mobile (iOS/Android)                                    40845
1527  Mobile (iOS/Android)                                    39970
1548  Mobile (iOS/Android)                                    13895
1551  Mobile (iOS/Android)                                    42490
1580                   iOS                                    50960
1582               Android                                     4970
1642                   iOS                                    51800
1644  Mobile (iOS/Android)                                    44380
1761  Mobile (iOS/Android)                                    17710
1807  Mobile (iOS/Android)                                    25830
1808  Mobile (iOS/Android)                                    19950
1835               Android                                    40040
1878  Mobile (iOS/Android)                                    52150
1979                   iOS                                    45360
2272                   iOS                                    33950
2337                   iOS                                    49735
2535  Mobile (iOS/Android)                                    28076
2709                   iOS                                     5705
2869  Mobile (iOS/Android)                                    20090
2934  Mobile (iOS/Android)                                    40845
3307  Mobile (iOS/Android)                                    36642
4229               Android                                    46810
4538  Mobile (iOS/Android)                                    20826
5861  Mobile (iOS/Android)                                    12108

      yearly_avg_Outstation_checkins  member_in_family  \
57                                23                 4
78                                 1                 4
81                                 1                 3
110                                1                 4
112                               26                 3
172                                1                 2
174                                1                 2
291                                1                 3
365                                1                 4
408                                1                 2
509                                1                 3
802                                1                 3
867                                1                 3
1065                               1                 4
1239                              24                 1
1399                              11                 3
1464                               1                 3
1527                              23                 4
1548                               1                 4
1551                               1                 3
1580                               1                 4
1582                              26                 3
1642                               1                 2
1644                               1                 2
1761                            <NA>                 3
1807                               1                 5
1808                               1                 3
1835                               1                 4
1878                               1                 2
1979                               1                 3
2272                               1                 3
2337                               1                 3
2535                               1                 4
2709                              24                 1
2869                              11                 3
2934                               1                 3
3307                              10                 3
4229                               1                 3
4538                               1                 1
5861                              10                 2

     preferred_location_type  Yearly_avg_comment_on_travel_page  \
57                   Medical                                  3
78                   Medical                                  3
81                   Medical                                  3
110                  Medical                                  3
112          Tour and Travel                                  3
172                  Medical                                  3
174                Financial                                  3
291            Entertainment                                  3
365                  Medical                                  3
408                Financial                                  3
509                Financial                                  3
802                Financial                                  3
867                  Medical                                  3
1065               Financial                                  3
1239           Entertainment                                  3
1399               Financial                                  3
1464                   Other                                  3
1527                 Medical                                  3
1548                 Medical                                  3
1551                 Medical                                  3
1580                 Medical                                  3
1582         Tour and Travel                                  3
1642                 Medical                                  3
1644               Financial                                  3
1761           Entertainment                                  3
1807                   Other                                  3
1808            Social media                                  3
1835                 Medical                                  3
1878               Financial                                  3
1979               Financial                                  3
2272               Financial                                  3
2337                 Medical                                  3
2535               Financial                                  3
2709           Entertainment                                  3
2869               Financial                                  3
2934                   Other                                  3
3307           Entertainment                                615
4229         Tour and Travel                                215
4538           Entertainment                                815
5861               Financial                                685

      total_likes_on_outofstation_checkin_received  \
57                                            4014
78                                           13664
81                                            2859
110                                           7484
112                                          17328
172                                           2088
174                                           5006
291                                           4485
365                                           5258
408                                          16555
509                                           7725
802                                           4302
867                                          17856
1065                                          4035
1239                                          5238
1399                                          7510
1464                                          2966
1527                                          4014
1548                                         13664
1551                                          2859
1580                                          7484
1582                                         17328
1642                                          2088
1644                                          5006
1761                                          4485
1807                                          3983
1808                                          6118
1835                                          5258
1878                                         16555
```

```
1979                                 7725
2272                                 4302
2337                                17856
2535                                 4035
2789                                 5238
2869                                 7510
2934                                 2966
3387                                10536
4229                                 2644
4538                                 2575
5861                                 5392

      week_since_last_outstation_checkin following_company_page  \
57                                     3                    Yes
78                                     4                     No
81                                     1                     No
110                                    3                     No
112                                    2                    Yes
172                                    4                     No
174                                    4                    Yes
291                                    4                     No
365                                    2                     No
408                                    2                    nan
509                                    3                     No
802                                    0                     No
867                                    2                     No
1065                                   0                    Yes
1239                                   2                     No
1399                                   1                     No
1464                                   0                     No
1527                                   3                    Yes
1548                                   4                     No
1551                                   1                     No
1580                                   3                     No
1582                                   2                    Yes
1642                                   4                     No
1644                                   4                    Yes
1761                                   4                     No
1807                                   0                     No
1808                                   1                     No
1835                                   2                     No
1878                                   2                     No
1979                                   3                     No
2272                                   0                     No
2337                                   2                     No
2535                                   0                    Yes
2789                                   2                     No
2869                                   1                     No
2934                                   0                     No
3387                                   6                     No
4229                                   6                     No
4538                                   2                     No
5861                                   3                     No

      montly_avg_comment_on_company_page working_flag  \
57                                    15           No
78                                    13           No
81                                    18           No
110                                   20          Yes
112                                   12           No
172                                   12           No
174                                   11           No
291                                   12           No
365                                   14           No
408                                   13           No
509                                   23          Yes
802                                   17           No
867                                   22          Yes
1065                                  16           No
1239                                  20          Yes
1399                                  17           No
1464                                  18           No
1527                                  15           No
1548                                  13           No
1551                                  18           No
1580                                  20          Yes
1582                                  12           No
1642                                  12           No
1644                                  11           No
1761                                  12           No
1807                                  17           No
1808                                  13           No
1835                                  14           No
1878                                  13           No
1979                                  23          Yes
2272                                  17           No
2337                                  22          Yes
2535                                  16           No
2789                                  20          Yes
2869                                  17           No
2934                                  18           No
3387                                  19           No
4229                                  22           No
4538                                  23           No
5861                                  14           No

      travelling_network_rating  Adult_flag  \
57                            3           1
78                            1           0
81                            1           0
110                           3           0
112                           3           0
172                           3           0
174                           1           1
291                           4           0
365                           3           0
408                           4           0
509                           3           1
802                           3           1
867                           3           1
1065                          2           0
1239                          4           0
1399                          2           1
1464                          4           0
1527                          3           1
1548                          1           0
1551                          1           0
1580                          3           0
1582                          3           0
1642                          3           0
1644                          1           1
1761                          4           0
1807                          3           0
1808                          3           3
1835                          3           0
1878                          4           0
1979                          3           1
2272                          3           1
2337                          3           1
2535                          2           0
2789                          4           0
2869                          2           1
2934                          4           0
3387                          4           0
4229                          4           3
4538                          3           1
5861                          3           1

      Daily_Avg_mins_spend_on_traveling_page
57                                         4
78                                        16
81                                         6
110                                       23
112                                       23
172                                       13
174                                        9
291                                       10
365                                        7
408                                       31
509                                       15
802                                        4
867                                       32
1065                                       4
1239                                       9
1399                                      10
1464                                       5
1527                                       4
1548                                      16
1551                                       6
1580                                      23
1582                                      23
1642                                      13
1644                                       9
1761                                      10
1807                                       4
1808                                      10
```

```
1835                                7
1878                               31
1979                               15
2272                                4
2337                               32
2535                                4
2709                                9
2869                               10
2934                                5
3307                               22
4229                               20
4538                                9
5861                               17
```

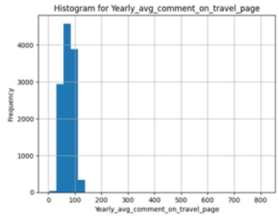
In [155…

```python
# Boxplot for detecting outliers
sns.boxplot(data=Data, y='Yearly_avg_comment_on_travel_page')
plt.title('Boxplot for Yearly_avg_comment_on_travel_page')
plt.show()
```



In [156…

```python
# Histogram for distribution
Data['Yearly_avg_comment_on_travel_page'].hist(bins=30)
plt.title('Histogram for Yearly_avg_comment_on_travel_page')
plt.xlabel('Yearly_avg_comment_on_travel_page')
plt.ylabel('Frequency')
plt.show()
```



In [157…

```python
from scipy.stats import zscore

# Calculate Z-scores
Data['z_score'] = zscore(Data['Yearly_avg_comment_on_travel_page'])
```

```python
# Filter outliers
outliers = Data[Data['z_score'].abs() > 3]
print(f"Outliers based on Z-score:\n", outliers)
```

```
Outliers based on Z-score:
         UserID Taken_product  Yearly_avg_view_on_travel_page  \
57    1000058            No                             262
78    1000079            No                             277
81    1000082            No                             262
110   1000111            No                             382
112   1000113            No                             405
172   1000173            No                             270
174   1000175            No                             280
291   1000292            No                             270
365   1000366            No                             345
408   1000409            No                             280
509   1000510            No                             247
802   1000803            No                             247
867   1000868            No                             375
1065  1001066            No                             412
1239  1001240            No                             232
1399  1001400            No                             322
1464  1001465            No                             195
1527  1001528            No                             262
1548  1001549            No                             277
1551  1001552            No                             262
1580  1001581            No                             382
1582  1001583            No                             405
1642  1001643            No                             270
1644  1001645            No                             280
1761  1001762            No                             270
1807  1001808            No                             217
1808  1001809            No                             225
1835  1001836            No                             345
1878  1001879            No                             280
1979  1001980            No                             247
2272  1002273            No                             247
2337  1002338            No                             375
2535  1002536            No                             412
2709  1002710            No                             232
2869  1002870            No                             322
2934  1002935            No                             195
3307  1003308            No                             377
4229  1004230            No                             286
4538  1004539            No                             280
5861  1005862            No                             287

          preferred_device  total_likes_on_outstation_checkin_given  \
57    Mobile (iOS/Android)                                    39970
78    Mobile (iOS/Android)                                    13895
81    Mobile (iOS/Android)                                    42490
110                    iOS                                    50960
112                Android                                     4970
172                    iOS                                    51800
174   Mobile (iOS/Android)                                    44380
291   Mobile (iOS/Android)                                    17710
365                Android                                    40040
408   Mobile (iOS/Android)                                    52150
509                    iOS                                    45360
802                    iOS                                    33950
867                    iOS                                    49735
1065  Mobile (iOS/Android)                                    28076
1239                   iOS                                     5705
1399  Mobile (iOS/Android)                                    20090
1464  Mobile (iOS/Android)                                    40845
1527  Mobile (iOS/Android)                                    39970
1548  Mobile (iOS/Android)                                    13895
1551  Mobile (iOS/Android)                                    42490
1580                   iOS                                    50960
1582               Android                                     4970
1642                   iOS                                    51800
1644  Mobile (iOS/Android)                                    44380
1761  Mobile (iOS/Android)                                    17710
1807  Mobile (iOS/Android)                                    25830
1808  Mobile (iOS/Android)                                    19950
1835               Android                                    40040
1878  Mobile (iOS/Android)                                    52150
1979                   iOS                                    45360
2272                   iOS                                    33950
2337                   iOS                                    49735
2535  Mobile (iOS/Android)                                    28076
2709                   iOS                                     5705
2869  Mobile (iOS/Android)                                    20090
2934  Mobile (iOS/Android)                                    40845
```

```
3307  Mobile (iOS/Android)                          36642
4229               Android                          46810
4538  Mobile (iOS/Android)                          20826
5861  Mobile (iOS/Android)                          12108

      yearly_avg_Outstation_checkins  member_in_family  \
57                                23                 4
78                                 1                 4
81                                 1                 3
110                                1                 4
112                               26                 3
172                                1                 2
174                                1                 2
291                                1                 3
365                                1                 4
408                                1                 2
509                                1                 3
802                                1                 3
867                                1                 3
1065                               1                 4
1239                              24                 1
1399                              11                 3
1464                               1                 3
1527                              23                 4
1548                               1                 4
1551                               1                 3
1580                               1                 4
1582                              26                 3
1642                               1                 2
1644                               1                 2
1761                            <NA>                 3
1807                               1                 5
1808                               1                 3
1835                               1                 4
1878                               1                 2
1979                               1                 3
2272                               1                 3
2337                               1                 3
2535                               1                 4
2709                              24                 1
2869                              11                 3
2934                               1                 3
3307                              10                 3
4229                               1                 3
4538                               1                 1
5861                              10                 2

     preferred_location_type  Yearly_avg_comment_on_travel_page  \
57                   Medical                                  3
78                   Medical                                  3
81                   Medical                                  3
110                  Medical                                  3
112          Tour and Travel                                  3
172                  Medical                                  3
174                Financial                                  3
291            Entertainment                                  3
365                  Medical                                  3
408                Financial                                  3
509                Financial                                  3
802                Financial                                  3
867                  Medical                                  3
1065               Financial                                  3
1239           Entertainment                                  3
1399               Financial                                  3
1464                   Other                                  3
1527                 Medical                                  3
1548                 Medical                                  3
1551                 Medical                                  3
1580                 Medical                                  3
1582         Tour and Travel                                  3
1642                 Medical                                  3
1644               Financial                                  3
1761           Entertainment                                  3
1807                   Other                                  3
1808            Social media                                  3
1835                 Medical                                  3
1878               Financial                                  3
1979               Financial                                  3
2272               Financial                                  3
2337                 Medical                                  3
2535               Financial                                  3
2709           Entertainment                                  3
2869               Financial                                  3
```

```
2934                   Other                                  3
3307           Entertainment                                615
4229         Tour and Travel                                215
4538           Entertainment                                815
5861               Financial                                685

      total_likes_on_outofstation_checkin_received  \
57                                            4014
78                                           13664
81                                            2859
110                                           7484
112                                          17328
172                                           2088
174                                           5006
291                                           4485
365                                           5258
408                                          16555
509                                           7725
802                                           4302
867                                          17856
1065                                          4035
1239                                          5238
1399                                          7510
1464                                          2966
1527                                          4014
1548                                         13664
1551                                          2859
1580                                          7484
1582                                         17328
1642                                          2088
1644                                          5006
1761                                          4485
1807                                          3983
1808                                          6118
1835                                          5258
1878                                         16555
1979                                          7725
2272                                          4302
2337                                         17856
2535                                          4035
2709                                          5238
2869                                          7510
2934                                          2966
3307                                         10536
4229                                          2644
4538                                          2575
5861                                          5392

      week_since_last_outstation_checkin following_company_page  \
57                                     3                    Yes
78                                     4                     No
81                                     1                     No
110                                    3                     No
112                                    2                    Yes
172                                    4                     No
174                                    4                    Yes
291                                    4                     No
365                                    2                     No
408                                    2                    nan
509                                    3                     No
802                                    0                     No
867                                    2                     No
1065                                   0                    Yes
1239                                   2                     No
1399                                   1                     No
1464                                   0                     No
1527                                   3                    Yes
1548                                   4                     No
1551                                   1                     No
1580                                   3                     No
1582                                   2                    Yes
1642                                   4                     No
1644                                   4                    Yes
1761                                   4                     No
1807                                   0                     No
1808                                   1                     No
1835                                   2                     No
1878                                   2                     No
1979                                   3                     No
2272                                   0                     No
2337                                   2                     No
2535                                   0                    Yes
2709                                   2                     No
```

```
2869                                   1              No
2934                                   0              No
3307
4229
4538
5861
```

To exit full screen, press Esc

```
      montly_avg_comment_on_company_page working_flag  \
57                                    15           No
78                                    13           No
81                                    18           No
110                                   20          Yes
112                                   12           No
172                                   12           No
174                                   11           No
291                                   12           No
365                                   14           No
408                                   13           No
509                                   23          Yes
802                                   17           No
867                                   22          Yes
1065                                  16           No
1239                                  20          Yes
1399                                  17           No
1464                                  18           No
1527                                  15           No
1548                                  13           No
1551                                  18           No
1580                                  20          Yes
1582                                  12           No
1642                                  12           No
1644                                  11           No
1761                                  12           No
1807                                  17           No
1808                                  13           No
1835                                  14           No
1878                                  13           No
1979                                  23          Yes
2272                                  17           No
2337                                  22          Yes
2535                                  16           No
2709                                  20          Yes
2869                                  17           No
2934                                  18           No
3307                                  19           No
4229                                  22           No
4538                                  23           No
5861                                  14           No

      travelling_network_rating  Adult_flag  \
57                            3           1
78                            1           0
81                            1           0
110                           3           0
112                           3           0
172                           3           0
174                           1           1
291                           4           0
365                           3           0
408                           4           0
509                           3           1
802                           3           1
867                           3           1
1065                          2           0
1239                          4           0
1399                          2           1
1464                          4           0
1527                          3           1
1548                          1           0
1551                          1           0
1580                          3           0
1582                          3           0
1642                          3           0
1644                          1           1
1761                          4           0
1807                          3           0
1808                          3           3
1835                          3           0
1878                          4           0
1979                          3           1
2272                          3           1
2337                          3           1
2535                          2           0
```

```
2709                          4           0
2869                          2           1
2934                          4           0
3307                          4           0
4229                          4           3
4538                          3           1
5861                          3           1

      Daily_Avg_mins_spend_on_traveling_page    z_score
57                                         4  -3.014735
78                                        16  -3.014735
81                                         6  -3.014735
110                                       23  -3.014735
112                                       23  -3.014735
172                                       13  -3.014735
174                                        9  -3.014735
291                                       10  -3.014735
365                                        7  -3.014735
408                                       31  -3.014735
509                                       15  -3.014735
802                                        4  -3.014735
867                                       32  -3.014735
1065                                       4  -3.014735
1239                                       9  -3.014735
1399                                      10  -3.014735
1464                                       5  -3.014735
1527                                       4  -3.014735
1548                                      16  -3.014735
1551                                       6  -3.014735
1580                                      23  -3.014735
1582                                      23  -3.014735
1642                                      13  -3.014735
1644                                       9  -3.014735
1761                                      10  -3.014735
1807                                       4  -3.014735
1808                                      10  -3.014735
1835                                       7  -3.014735
1878                                      31  -3.014735
1979                                      15  -3.014735
2272                                       4  -3.014735
2337                                      32  -3.014735
2535                                       4  -3.014735
2709                                       9  -3.014735
2869                                      10  -3.014735
2934                                       5  -3.014735
3307                                      22  22.684145
4229                                      20   5.887491
4538                                       9  31.082471
5861                                      17  25.623559
```

In [158…

```python
# Check for outliers in all numerical columns using IQR
numerical_columns = Data.select_dtypes(include=['int64', 'float64']).columns

for column in numerical_columns:
    outliers = detect_outliers_iqr(Data, column)
    print(f"Outliers in '{column}': {len(outliers)}")
```

```
Outliers in 'UserID': 0
Outliers in 'yearly_avg_Outstation_checkins': 0
Outliers in 'member_in_family': 11
Outliers in 'total_likes_on_outofstation_checkin_received': 916
Outliers in 'week_since_last_outstation_checkin': 0
Outliers in 'montly_avg_comment_on_company_page': 242
Outliers in 'travelling_network_rating': 0
Outliers in 'Adult_flag': 680
Outliers in 'Daily_Avg_mins_spend_on_traveling_page': 350
Outliers in 'z_score': 40
```

In [159…

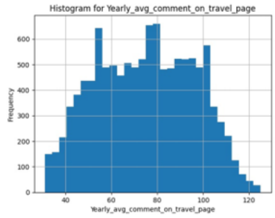
```python
# Calculate IQR and bounds for 'Yearly_avg_comment_on_travel_page'
Q1 = Data['Yearly_avg_comment_on_travel_page'].quantile(0.25)  # First quartile
Q3 = Data['Yearly_avg_comment_on_travel_page'].quantile(0.75)  # Third quartile
IQR = Q3 - Q1  # Interquartile range
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the data to remove outliers
Data = Data[(Data['Yearly_avg_comment_on_travel_page'] >= lower_bound) &
            (Data['Yearly_avg_comment_on_travel_page'] <= upper_bound)]

# Verify the changes
print(f"Data shape after removing outliers: {Data.shape}")
```

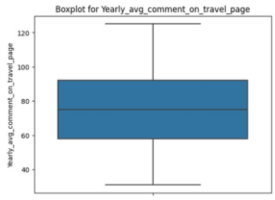
```
Data shape after removing outliers: (11720, 18)
```

univariate analysis and bivariate analysis

```
In [160…
# Descriptive statistics
print(Data.describe())

# Histogram
Data['Yearly_avg_comment_on_travel_page'].hist(bins=30)
plt.title('Histogram for Yearly_avg_comment_on_travel_page')
plt.xlabel('Yearly_avg_comment_on_travel_page')
plt.ylabel('Frequency')
plt.show()

# Boxplot
sns.boxplot(y=Data['Yearly_avg_comment_on_travel_page'])
plt.title('Boxplot for Yearly_avg_comment_on_travel_page')
plt.show()
```

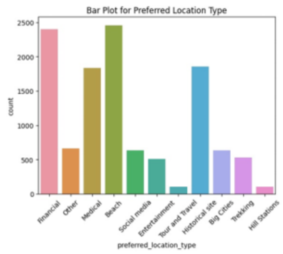
```
             UserID  Yearly_avg_view_on_travel_page  \
count  1.172000e+04                    11720.000000
mean   1.005895e+06                      280.741468
std    3.390781e+03                       66.489706
min    1.000001e+06                       35.000000
25%    1.002967e+06                      233.000000
50%    1.005900e+06                      275.000000
75%    1.008830e+06                      322.000000
max    1.011760e+06                      464.000000

       total_likes_on_outstation_checkin_given  \
count                             11720.000000
mean                              28150.379693
std                               14142.983307
min                                3570.000000
25%                               16695.000000
50%                               28076.000000
75%                               40110.000000
max                              252430.000000

       yearly_avg_Outstation_checkins  member_in_family  \
count                         11645.0           11720.0
mean                          8.23246           2.92116
std                           8.67156          1.045315
min                               1.0               1.0
25%                               1.0               2.0
50%                               4.0               3.0
75%                              14.0               4.0
max                              29.0              10.0

       Yearly_avg_comment_on_travel_page  \
count                       11720.000000
mean                           74.840956
std                            21.169823
min                            31.000000
25%                            58.000000
50%                            75.000000
75%                            92.000000
max                           125.000000

       total_likes_on_outofstation_checkin_received  \
count                                  11720.000000
mean                                    6529.418686
std                                     4705.578418
min                                     1009.000000
25%                                     2938.000000
50%                                     4947.000000
75%                                     8394.250000
max                                    20065.000000

       week_since_last_outstation_checkin  montly_avg_comment_on_company_page  \
count                        11720.000000                        11720.000000
mean                             3.206655                           28.703584
std                              2.618657                           48.737672
min                              0.000000                           11.000000
25%                              1.000000                           17.000000
50%                              3.000000                           22.000000
75%                              5.000000                           27.000000
max                             11.000000                          500.000000

       travelling_network_rating    Adult_flag  \
count               11720.000000  11720.000000
mean                    2.711775      0.794881
std                     1.081237      0.852001
min                     1.000000      0.000000
25%                     2.000000      0.000000
50%                     3.000000      1.000000
75%                     4.000000      1.000000
max                     4.000000      3.000000

       Daily_Avg_mins_spend_on_traveling_page       z_score
count                            11720.000000  11720.000000
mean                                13.819881      0.001984
std                                  9.072250      0.888955
min                                  0.000000     -1.838969
25%                                  8.000000     -0.705195
50%                                 12.000000      0.008663
75%                                 18.000000      0.722520
max                                270.000000      2.108244
```



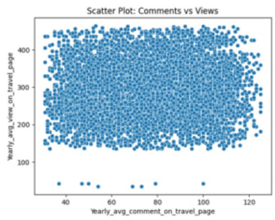



```python
In [161…
# Frequency table
print(Data['preferred_location_type'].value_counts())

# Bar plot
sns.countplot(x=Data['preferred_location_type'])
plt.title('Bar Plot for Preferred Location Type')
plt.xticks(rotation=45)
plt.show()
```

```
Beach               2455
Financial           2396
Historical site     1856
Medical             1831
Other                664
Big Cities           636
Social media         632
Trekking             528
Entertainment        510
Hill Stations        108
Tour and Travel      104
Name: preferred_location_type, dtype: int64
```



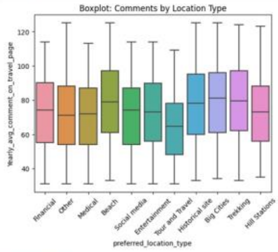

```python
In [162…
# Scatter plot
sns.scatterplot(x=Data['Yearly_avg_comment_on_travel_page'], y=Data['Yearly_avg_view_on_travel_page'])
plt.title('Scatter Plot: Comments vs Views')
plt.xlabel('Yearly_avg_comment_on_travel_page')
plt.ylabel('Yearly_avg_view_on_travel_page')
plt.show()

# Correlation
correlation = Data[['Yearly_avg_comment_on_travel_page', 'Yearly_avg_view_on_travel_page']].corr()
print("Correlation Matrix:\n", correlation)
```



```
Correlation Matrix:
                              Yearly_avg_comment_on_travel_page  \
Yearly_avg_comment_on_travel_page                      1.000000
Yearly_avg_view_on_travel_page                         0.044778

                              Yearly_avg_view_on_travel_page
Yearly_avg_comment_on_travel_page                   0.044778
Yearly_avg_view_on_travel_page                      1.000000
```

In [163…

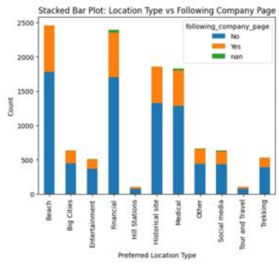
```
# Boxplot
sns.boxplot(x=Data['preferred_location_type'], y=Data['Yearly_avg_comment_on_travel_page'])
plt.title('Boxplot: Comments by Location Type')
plt.xticks(rotation=45)
plt.show()
```



In [164…

```
# Cross-tabulation
crosstab = pd.crosstab(Data['preferred_location_type'], Data['following_company_page'])
print(crosstab)

# Stacked bar plot
crosstab.plot(kind='bar', stacked=True)
plt.title('Stacked Bar Plot: Location Type vs Following Company Page')
plt.xlabel('Preferred Location Type')
plt.ylabel('Count')
plt.show()
```

```
following_company_page     No  Yes  nan
preferred_location_type
Beach                    1780  675    0
Big Cities                444  192    0
Entertainment             372  128   10
Financial                1704  645   47
Hill Stations              76   32    0
Historical site          1328  528    0
Medical                  1284  515   32
Other                     441  218    5
Social media              434  190    8
Tour and Travel            74   30    0
Trekking                  392  136    0
```



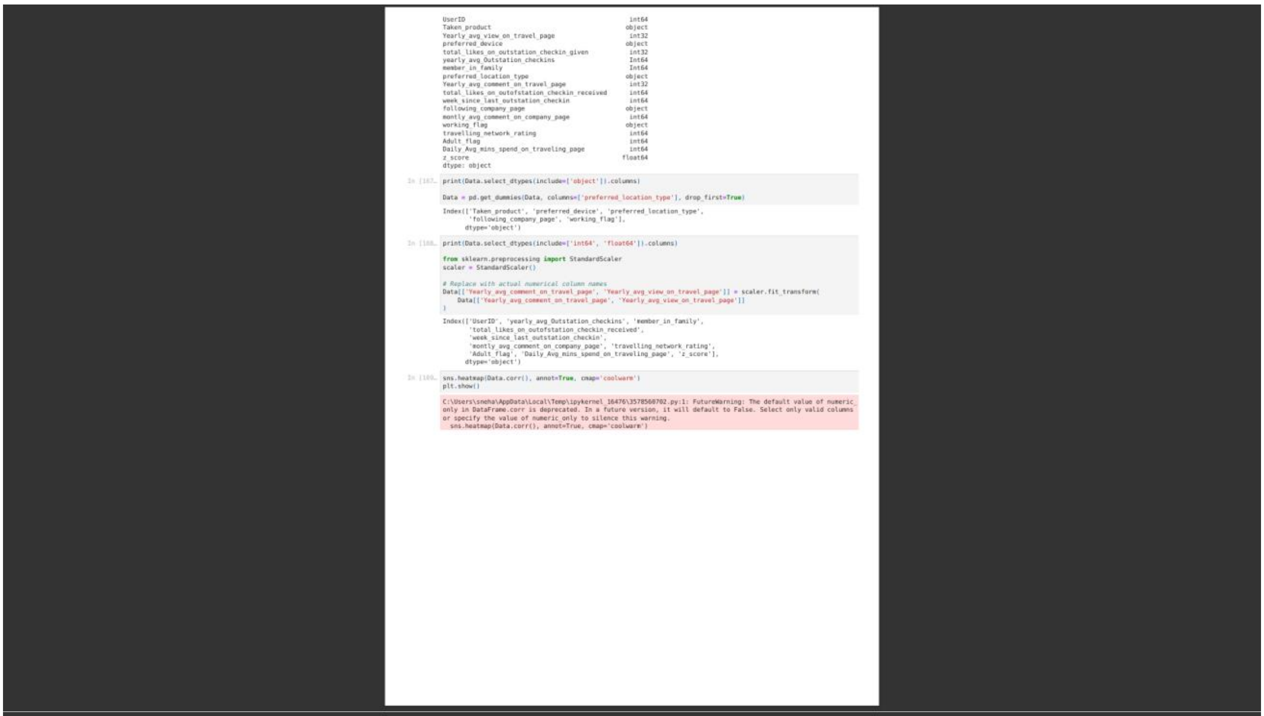
In [165…

```
print(Data.isnull().sum())
```

```
UserID                                        0
Taken_product                                 0
Yearly_avg_view_on_travel_page                0
preferred_device                             53
total_likes_on_outstation_checkin_given       0
yearly_avg_Outstation_checkins               75
member_in_family                              0
preferred_location_type                       0
Yearly_avg_comment_on_travel_page             0
total_likes_on_outofstation_checkin_received  0
week_since_last_outstation_checkin            0
following_company_page                        0
montly_avg_comment_on_company_page            0
working_flag                                  0
travelling_network_rating                     0
Adult_flag                                    0
Daily_Avg_mins_spend_on_traveling_page        0
z_score                                       0
dtype: int64
```

In [166…

```
print(Data.dtypes)
```



```
UserID                                         int64
Taken_product                                 object
Yearly_avg_view_on_travel_page                 int32
preferred_device                              object
total_likes_on_outstation_checkin_given        int32
yearly_avg_Outstation_checkins                 Int64
member_in_family                               Int64
preferred_location_type                       object
Yearly_avg_comment_on_travel_page              int32
total_likes_on_outofstation_checkin_received   int64
week_since_last_outstation_checkin             int64
following_company_page                        object
montly_avg_comment_on_company_page             int64
working_flag                                  object
travelling_network_rating                      int64
Adult_flag                                     int64
Daily_Avg_mins_spend_on_traveling_page         int64
z_score                                      float64
dtype: object
```

```
print(Data.select_dtypes(include=['object']).columns)

Data = pd.get_dummies(Data, columns=['preferred_location_type'], drop_first=True)
```

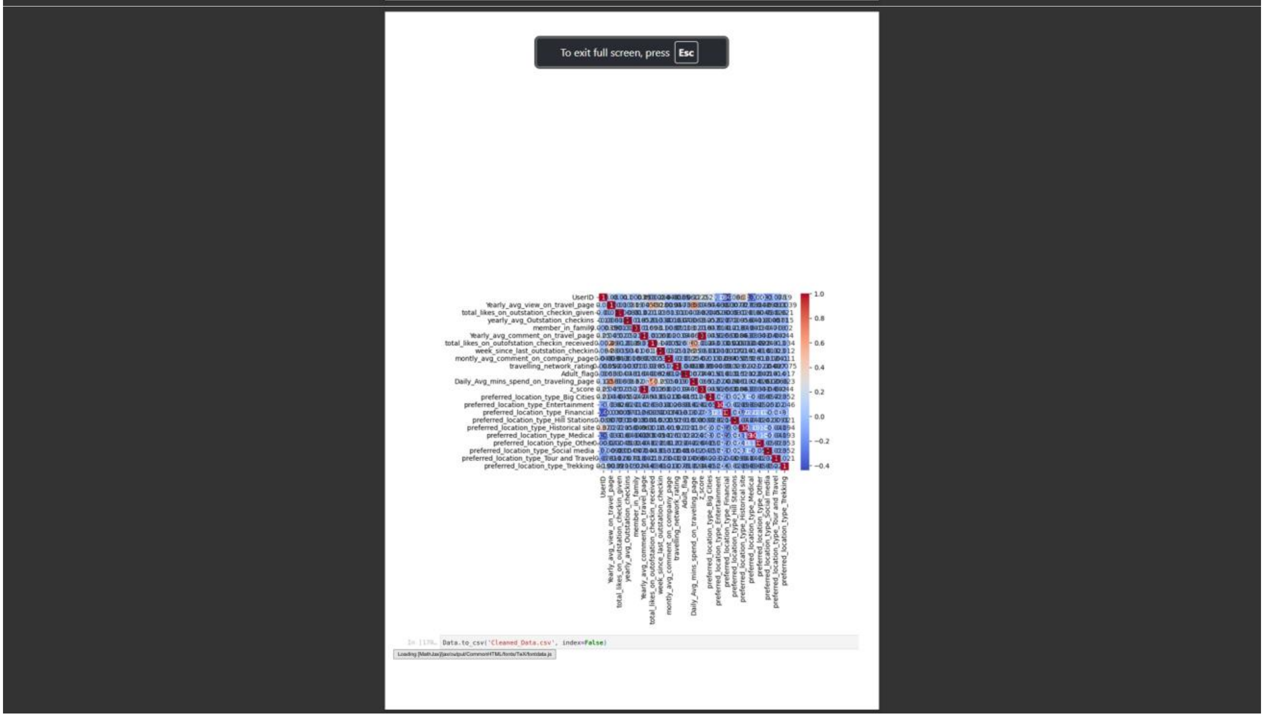
```
Index(['Taken_product', 'preferred_device', 'preferred_location_type',
       'following_company_page', 'working_flag'],
      dtype='object')
```

```
print(Data.select_dtypes(include=['int64', 'float64']).columns)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# Replace with actual numerical column names
Data[['Yearly_avg_comment_on_travel_page', 'Yearly_avg_view_on_travel_page']] = scaler.fit_transform(
    Data[['Yearly_avg_comment_on_travel_page', 'Yearly_avg_view_on_travel_page']]
)
```

```
Index(['UserID', 'yearly_avg_Outstation_checkins', 'member_in_family',
       'total_likes_on_outofstation_checkin_received',
       'week_since_last_outstation_checkin',
       'montly_avg_comment_on_company_page', 'travelling_network_rating',
       'Adult_flag', 'Daily_Avg_mins_spend_on_traveling_page', 'z_score'],
      dtype='object')
```

```
sns.heatmap(Data.corr(), annot=True, cmap='coolwarm')
plt.show()
```

```
C:\Users\sneha\AppData\Local\Temp\ipykernel_16476\3578560702.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(Data.corr(), annot=True, cmap='coolwarm')
```



Model Building: [Model Building Link]

In [7]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [8]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

# 1. Load data
data = pd.read_csv('D:\\MBA\\Semester 4\\Project\\Data(TM)\\Python implementatio

# 2. Feature selection (replace with your actual feature columns)
feature_cols = [
    'Yearly_avg_comment_on_travel_page',
    'Yearly_avg_view_on_travel_page',
    'total_likes_on_outstation_checkin_given',
    'yearly_avg_Outstation_checkins',
    'member_in_family',
    # add more relevant features as needed
]
X = data[feature_cols]
y = data['Taken_product']  # updated target variable

# 3. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# 4. Model training
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# 5. Evaluation
y_pred = model.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

          No       0.84      1.00      0.91      1968
         Yes       0.00      0.00      0.00       376

    accuracy                           0.84      2344
   macro avg       0.42      0.50      0.46      2344
weighted avg       0.70      0.84      0.77      2344

Confusion Matrix:
 [[1968    0]
 [ 376    0]]
```

```
c:\Users\sneha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-def
ined and being set to 0.0 in labels with no predicted samples. Use `zero_divisi
on` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\sneha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-def
ined and being set to 0.0 in labels with no predicted samples. Use `zero_divisi
on` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\sneha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklear
n\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-def
ined and being set to 0.0 in labels with no predicted samples. Use `zero_divisi
on` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

As of now the Data is imbalanced

In [9]:
```python
print(data['Taken_product'].value_counts())
```

```
No     9824
Yes    1896
Name: Taken_product, dtype: int64
```

In [10]:
```python
print(np.unique(y_pred, return_counts=True))
```

```
(array(['No'], dtype=object), array([2344], dtype=int64))
```

In [11]:
```python
model = LogisticRegression(max_iter=1000, class_weight='balanced')
model.fit(X_train, y_train)
```

Out[11]:
```
LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=1000)
```

In [16]:
```python
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
model.fit(X_resampled, y_resampled)
```

Out[16]:
```
LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=1000)
```

In [15]:
```python
pip install imbalanced-learn
```

```
  WARNING: Failed to remove contents in a temporary directory 'C:\Users\sneha\AppData\Local\Programs\Python\Python310\Lib\site-packages\~umpy'.
  You can safely remove it manually.

[notice] A new release of pip is available: 25.0.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [17]:
```
from sklearn.metrics import classification_report, confusion_matrix

print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

          No       0.84      1.00      0.91      1968
         Yes       0.00      0.00      0.00       376

    accuracy                           0.84      2344
   macro avg       0.42      0.50      0.46      2344
weighted avg       0.70      0.84      0.77      2344

Confusion Matrix:
 [[1968    0]
 [ 376    0]]
```

```
c:\Users\sneha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\sneha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
c:\Users\sneha\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Your model is still predicting **only the majority class ("No")** for all test samples, even after using SMOTE. Hence trying to check with different model

In [18]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(class_weight='balanced', random_state=42)
rf.fit(X_resampled, y_resampled)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test, y_pred_rf))
print(confusion_matrix(y_test, y_pred_rf))
```

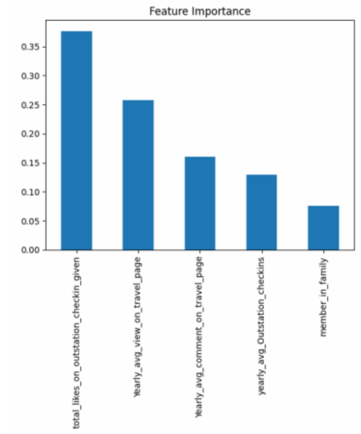
```
              precision    recall  f1-score   support

          No       0.98      0.98      0.98      1968
         Yes       0.92      0.89      0.90       376

    accuracy                           0.97      2344
   macro avg       0.95      0.94      0.94      2344
weighted avg       0.97      0.97      0.97      2344

[[1937   31]
 [  41  335]]
```

This output shows that your **Random Forest model** is performing much better than Logistic Regression for the imbalanced data:

- Random Forest model is able to **detect both classes well**, including the minority class ("Yes").
- The model is **not biased toward the majority class** anymore.
- **Precision and recall for both classes are high**, indicating a balanced and effective model.

In [19]:
```
importances = rf.feature_importances_
feature_importance = pd.Series(importances, index=feature_cols).sort_values(asce
print(feature_importance)
feature_importance.plot(kind='bar')
plt.title('Feature Importance')
plt.show()
```

```
total_likes_on_outstation_checkin_given    0.376483
Yearly_avg_view_on_travel_page             0.257311
Yearly_avg_comment_on_travel_page          0.160333
yearly_avg_Outstation_checkins             0.129739
member_in_family                           0.076135
dtype: float64
```



Each bar represents how much a particular feature contributed to the model's decision-making process for predicting whether a user took the product. Features with higher importance scores had a greater impact on the model's predictions, helping to identify which aspects of user engagement and behavior are most influential in driving ticket purchase decisions. This insight can guide airlines to focus their marketing efforts on the most impactful engagement metrics.

In [20]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Train the Decision Tree model on the resampled data
dt = DecisionTreeClassifier(class_weight='balanced', random_state=42)
dt.fit(X_resampled, y_resampled)

# Predict on the test set
y_pred_dt = dt.predict(X_test)
```

```
# Evaluate the model
print("Classification Report:\n", classification_report(y_test, y_pred_dt))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
```

```
Classification Report:
               precision    recall  f1-score   support

          No       0.98      0.97      0.97      1968
         Yes       0.84      0.88      0.86       376

    accuracy                           0.95      2344
   macro avg       0.91      0.92      0.92      2344
weighted avg       0.95      0.95      0.95      2344

Confusion Matrix:
 [[1905   63]
 [  46  330]]
```

- The Decision Tree model is **performing very well** for both classes.
- It is able to **detect the minority class ("Yes")** with high recall (0.88) and good precision (0.84).
- The model is **not biased toward the majority class** and provides a balanced performance.
- **Overall accuracy is high (95%)**, indicating strong predictive power.

- **Overall Performance:**
  Both models perform very well, but the **Random Forest** has slightly higher accuracy, precision, recall, and F1-score for both classes, especially for the minority class ("Yes").
- **Minority Class Detection:**
  - **Random Forest** detects more true positives for "Yes" (335 vs. 330) and has fewer false negatives (41 vs. 46) compared to the Decision Tree.
  - **Precision and recall for "Yes"** are higher in Random Forest, meaning it is better at correctly identifying users who took the product.
- **Generalization:**
  - **Random Forest** is an ensemble of many decision trees, which helps it generalize better and avoid overfitting.
  - **Decision Tree** is more prone to overfitting and may not perform as well on unseen data.
- **Interpretability:**
  - **Decision Tree** is easier to interpret and visualize.
  - **Random Forest** is less interpretable but provides more robust and reliable predictions.

Conclusion

- **Random Forest** is the better model for your data, offering higher accuracy and better detection of both classes, especially the minority class.
- **Decision Tree** is still a strong model and useful for understanding feature splits, but for deployment or business decisions, Random Forest is preferred due to its superior performance.

Model Validation

In [21]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(rf, X, y, cv=5, scoring='f1_weighted')
print("Cross-validated F1 scores:", scores)
print("Mean F1 score:", scores.mean())
```

```
Cross-validated F1 scores: [0.93504994 0.96312288 0.98216236 0.98030489 0.94831
859]
Mean F1 score: 0.9617917300718826
```

cross-validated F1 scores show that your Random Forest model is performing **consistently well** across different splits of your data:

Analysis

- **High and Stable Scores:**
  All F1 scores are above 0.93, with a mean of ~0.96. This indicates your model is robust and not overfitting to any particular fold.
- **Low Variance:**
  The scores do not fluctuate much, showing your model generalizes well to unseen data.
- **F1 Score:**
  The F1 score balances precision and recall, which is especially important for imbalanced datasets like yours.

In [22]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'class_weight': ['balanced']
}

# Initialize the model
rf = RandomForestClassifier(random_state=42)

# Set up GridSearchCV
grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    scoring='f1_weighted',
    n_jobs=-1,
```

```
    verbose=2
)

# Fit on the resampled training data
grid_search.fit(X_resampled, y_resampled)

# Best parameters and score
print("Best Parameters:", grid_search.best_params_)
print("Best F1 Score:", grid_search.best_score_)

# Use the best estimator to predict on the test set
best_rf = grid_search.best_estimator_
y_pred_best = best_rf.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred_best))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_best))
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
Best Parameters: {'class_weight': 'balanced', 'max_depth': None, 'min_samples_l
eaf': 1, 'min_samples_split': 2, 'n_estimators': 200}
Best F1 Score: 0.9749222973821372
Classification Report:
              precision    recall  f1-score   support

          No       0.98      0.98      0.98      1968
         Yes       0.92      0.89      0.91       376

    accuracy                           0.97      2344
   macro avg       0.95      0.94      0.94      2344
weighted avg       0.97      0.97      0.97      2344

Confusion Matrix:
 [[1938   30]
 [  40  336]]
```

The tuned Random Forest model maintains excellent performance on both classes, especially the minority class ("Yes"). High precision and recall for both classes indicate the model is both accurate and reliable. The model is well-balanced and generalizes well, as shown by the high cross-validated F1 score and strong test results.

In [23]:
```
import joblib

# Save the model
joblib.dump(best_rf, 'random_forest_model.pkl')

# To load the model later:
# loaded_model = joblib.load('random_forest_model.pkl')
```

Out[23]: `['random_forest_model.pkl']`

In [24]:
```
joblib.dump(smote, 'smote_object.pkl')
```

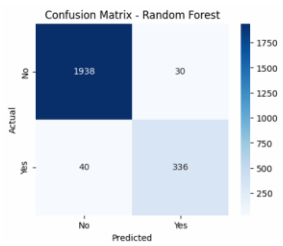
Out[24]: `['smote_object.pkl']`

In [26]:
```
import matplotlib.pyplot as plt

class_counts = data['Taken_product'].value_counts()
plt.figure(figsize=(6,4))
class_counts.plot(kind='bar', color=['skyblue', 'salmon'])
```

```
plt.title('Class Distribution in the Dataset')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```

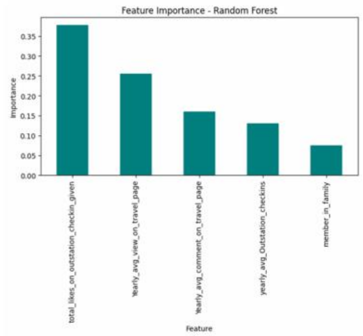
The dataset is highly imbalanced, with a much larger number of "No" responses compared to "Yes". This imbalance can negatively impact model performance, making it important to apply techniques like SMOTE to ensure both classes are learned effectively.

In [27]:
```
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_best)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No', 'Yes'], yt
plt.title('Confusion Matrix - Random Forest')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



The Random Forest model performs very well, with high precision and recall for both classes. The confusion matrix shows that the model correctly identifies most "Yes" and "No" cases, with very few misclassifications. This indicates the model is reliable for predicting product uptake.

In [28]:
```
feature_importance = pd.Series(best_rf.feature_importances_, index=feature_cols)
plt.figure(figsize=(8,4))
feature_importance.plot(kind='bar', color='teal')
plt.title('Feature Importance - Random Forest')
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.show()
```



The "total likes on outstation check-ins given" and "Yearly average view on travel page" are the most influential features in predicting product uptake. This insight can help the airline focus its marketing efforts on customers who are highly engaged with these activities.