Applied Project Report
Analysis of Real-life Business Cases

By

Maharshi Trivedi

A Master's Project Report submitted to Scaler Neovarsity - Woolf in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Month of Submission- October 2024

Scaler Mentee Email ID: maharshi.nit@gmail.com

Thesis Supervisor: Shivank Agarwal

Date of Submission: 16/10/2024

Certification

I confirm that I have overseen / reviewed this applied project and, in my judgment, it adheres to the appropriate standards of academic presentation. I believe it satisfactorily meets the criteria, in terms of both quality and breadth, to serve as an applied project report for the attainment of Master of Science in Computer Science degree. This applied project report has been submitted to Woolf and is deemed sufficient to fulfill the prerequisites for the Master of Science in Computer Science degree.

Shivank Agarwal

……

Project Guide / Supervisor

DECLARATION

I confirm that this project report, submitted to fulfill the requirements for the Master of Science in Computer Science degree, completed by me from June 2023 to October 2024, is the result of my own individual endeavor. The Project has been made on my own under the guidance of my supervisor with proper acknowledgement and without plagiarism. Any contributions from external sources or individuals, including the use of AI tools, are appropriately acknowledged through citation. By making this declaration, I acknowledge that any violation of this statement constitutes academic misconduct. I understand that such misconduct may lead to expulsion from the program and/or disqualification from receiving the degree.

Maharshi Trivedi

Signature of the Candidate **Date: 16 October 2024**

ACKNOWLEDGMENT

I would like to take a moment to express my deepest gratitude to my family, whose unwavering support and encouragement have been the foundation of my success. To my Scaler instructors, thank you for your guidance, expertise, and dedication throughout this journey—your insights and mentorship have been invaluable. Lastly, I am incredibly thankful to everyone who has inspired or motivated me along the way; your words and actions fueled my determination and played a crucial role in helping me reach this milestone. This accomplishment is as much yours as it is mine.

Table of Contents

List of Tables

(To be written sequentially as they appear in the text)

List of Figures

(List of Images, Graphs, Charts sequentially as they appear in the text)

Applied Real-life Business Cases

ABSTRACT

1. Five actual business case studies that were finished during a Data Science and Machine Learning (DSML) course are presented in this chapter. Every example shows how important DSML approaches are for streamlining corporate operations, raising profitability, and improving decision-making. The incidents, which have all been anonymized for privacy reasons, span several industries, including marketing, retail, e-commerce, financial services, and transportation.

2. Predictive modeling was used in the retail industry to estimate sales trends, which enhanced inventory control and decreased overstock expenses. An in-depth examination of user attrition on an e-commerce platform revealed important demographic elements that support retention tactics and boost client loyalty. A financial institution segmented its customer base using clustering algorithms, which enhanced client acquisition and allowed for more focused marketing campaigns. The transportation case study concentrated on using machine learning to optimize routes, which resulted in a considerable reduction in operating costs and an improvement in delivery times. Lastly, DSML methodologies were applied to the marketing domain to assess the efficacy of offline and online campaigns, maximizing budget allocation and producing increased returns on investment.

3. These case studies show how DSML technologies may be used in a variety of sectors to improve profitability, streamline operations, and make data-driven business choices. These tools include predictive analytics, time-series analysis, and machine learning algorithms.

Chapter 1: Business Case Study 1- SQL

Problem Description:

1. A well-known brand is a significant retailer in the US and has operations on a global scale. By providing unmatched value, creativity, innovation, and extraordinary customer experience that no other retailer can match, this company establishes itself as a preferred shopping destination.

2. The operations of the company in Brazil are the subject of this business case, which offers analytical data on 100,000 orders placed between 2016 and 2018. The dataset provides a detailed look at several variables, such as order status, pricing, payment and freight performance, customer geography, product features, and customer reviews.

3. This large dataset can be analyzed to reveal important information on the company's Brazilian operations. The data can provide insight into several business-related topics, including order fulfilment, pricing tactics, the effectiveness of payments and shipping, client demographics, product features, and customer satisfaction levels.

4. Assuming you are a data analyst/ scientist at Target, you have been assigned the task of analyzing the given dataset to extract valuable insights and provide actionable recommendations.

Business Questions to be answered from Analysis:

1. Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:
    1. Data type of all columns in the "customers" table.
    2. Getting the time range between which the orders were placed.
    3. Counting the Cities & States of customers who ordered during the given period.

2. In-depth Exploration:
    1. Is there a growing trend in the no. of orders placed over the past years?
    2. Can I see some kind of monthly seasonality in terms of the no. of orders being placed?
    3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

        1. 0-6 hrs.: Dawn
        2. 7-12 hrs.: Mornings
        3. 13-18 hrs.: Afternoon
        4. 19-23 hrs.: Night

3. Evolution of E-commerce orders in the Brazil region:
    1. Getting the month-on-month no. of orders placed in each state.
    2. How are the customers distributed across all the states?

4. Impact on Economy: Analyzed the money movement by e-commerce by looking at order prices, freight and others.
    1. Calculated the % increase in the cost of orders from year 2017 to 2018 (included months between Jan to Aug only). I have used the "payment_value" column in the payments table to get the cost of orders.
    2. Calculated the Total & Average value of order price for each state.
    3. Calculated the Total & Average value of order freight for each state.

5. Analysis based on sales, freight and delivery time:
    1. Finding the number of days taken to deliver each order from the order's purchase date as delivery time. Also, calculating the difference (in days) between the estimated & actual delivery date of an order. Did this in a single query. Calculating the delivery time and the difference between the estimated & actual delivery date using the given formula:
        1. time_to_deliver = order_delivered_customer_date - order_purchase_timestamp
        2. diff_estimated_delivery = order_delivered_customer_date - order_estimated_delivery_date

    2. Finding out the top 5 states with the highest & lowest average freight value.
    3. Finding out the top 5 states with the highest & lowest average delivery time.
    4. Finding out the top 5 states where the order delivery is fast as compared to the estimated date of delivery. Used the difference between the averages of actual & estimated delivery dates to figure out how fast the delivery was for each state.

6. Analysis based on the payments:
    1. Finding the month-on-month no. of orders placed using different payment types.
    2. Finding the no. of orders placed based on the payment installments that have been paid.

Methodology/Approach to Solve the Case Study:

1. Exploratory Data Analysis (EDA):
    1. Approach: Begin by exploring the structure of the dataset, checking for data types, null values, and basic statistics (mean, median, etc.) to understand the overall characteristics of the data.
    2. Application: EDA helps identify trends, outliers, and patterns that give an initial overview of business performance. It is essential for uncovering hidden insights and providing a basis for deeper analysis.

2. Geographical Analysis:
    1. Approach: Perform state and city-level analysis to understand customer distribution and geographic performance. Visualize data on maps to spot trends.
    2. Application: This helps businesses allocate resources, tailor marketing campaigns, and optimize logistics based on regional performance. It also assists in identifying potential expansion opportunities or areas needing improvement.

3. Customer Behavior Analysis:
    1. Approach: Segment customers based on factors like order timing, frequency, and payment methods. Use clustering techniques to group customers with similar purchasing behaviors.
    2. Application: Understanding customer behavior allows companies to personalize marketing, improve customer experience, and increase retention through targeted loyalty programs or promotions.

4. Logistic and Supply Chain Optimization:
    1. Approach: Analyze delivery times, freight costs, and logistics efficiency across states to identify potential optimizations.
    2. Application: By optimizing shipping routes, warehouse locations, and delivery processes, businesses can reduce costs, improve customer satisfaction, and increase overall efficiency in their supply chain operations.

Analysis:

Query 1:

```
#Q1: Import the dataset and do usual exploratory analysis steps like checking the structure & characteristics of the dataset:


-- 1. Data type of all columns in the "customers" table.
SELECT COLUMN_NAME, DATA_TYPE
FROM `casestudy-112.target.INFORMATION_SCHEMA.COLUMNS`
WHERE TABLE_NAME = 'customers';


-- 2. Get the time range between which the orders were placed.
WITH cte AS
(SELECT *,
EXTRACT (DATE FROM order_purchase_timestamp) AS order_date
FROM `casestudy-112.target.orders`
)
SELECT
DATE_DIFF(MAX(order_date), MIN(order_date), year) AS range_in_years,
DATE_DIFF(MAX(order_date), MIN(order_date), month) AS range_in_month,
DATE_DIFF(MAX(order_date), MIN(order_date), day) AS range_in_days
FROM cte;


-- 3. Count the number of Cities and States in our dataset.
SELECT COUNT(DISTINCT geolocation_city) AS number_of_cities, COUNT(DISTINCT geolocation_state) AS number_of_states
FROM `casestudy-112.target.geolocation`;
```

Figure 1.1: Importing the dataset and doing usual exploratory analysis

Output screenshots:

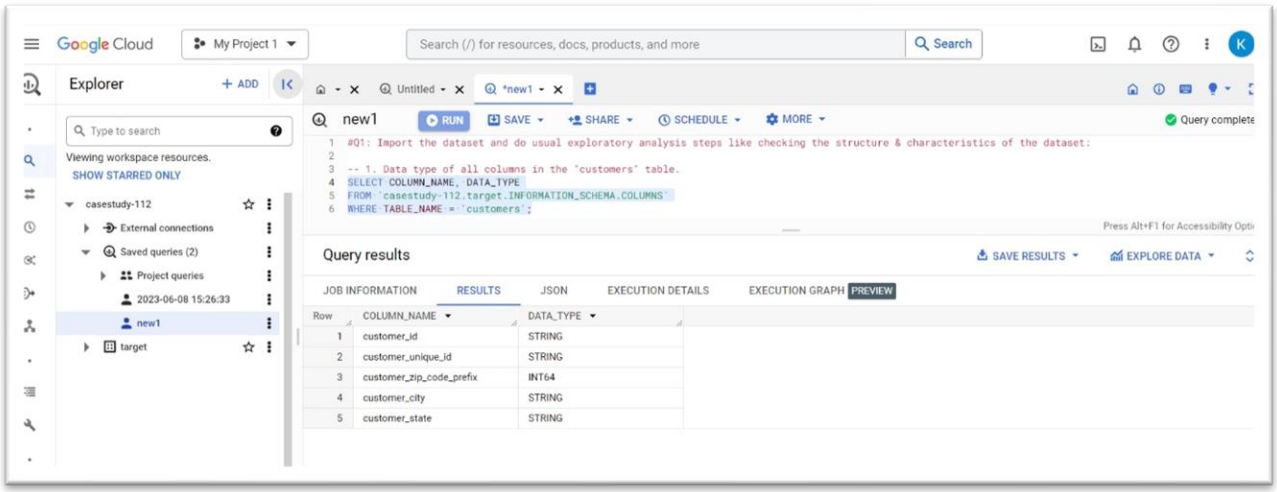


Figure 1.2: Data type of all columns in the "customers" table

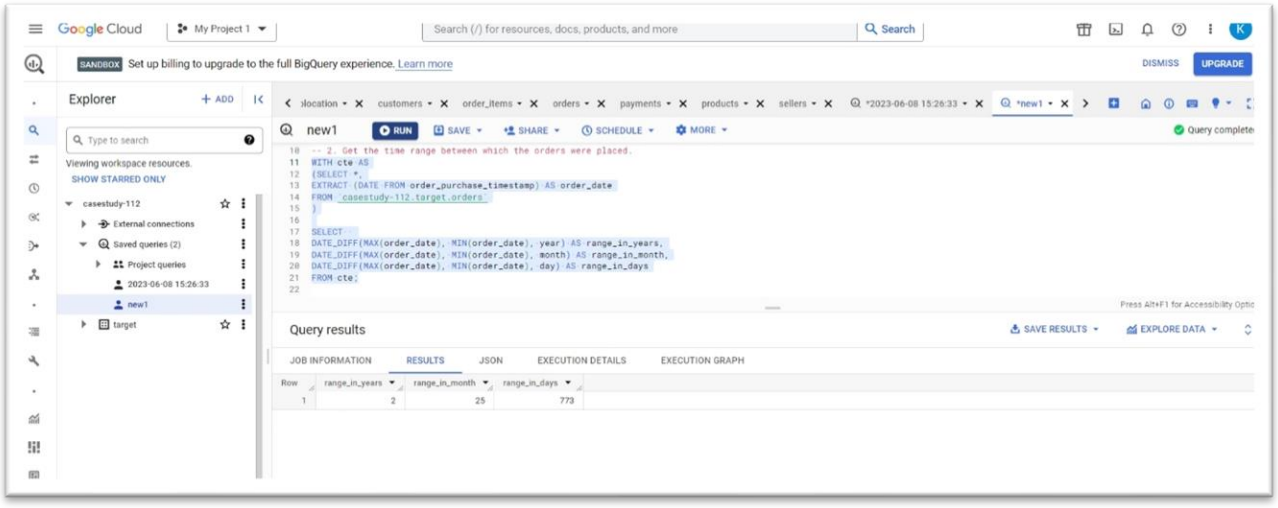


Figure 1.3: Getting the time range between which the orders were placed

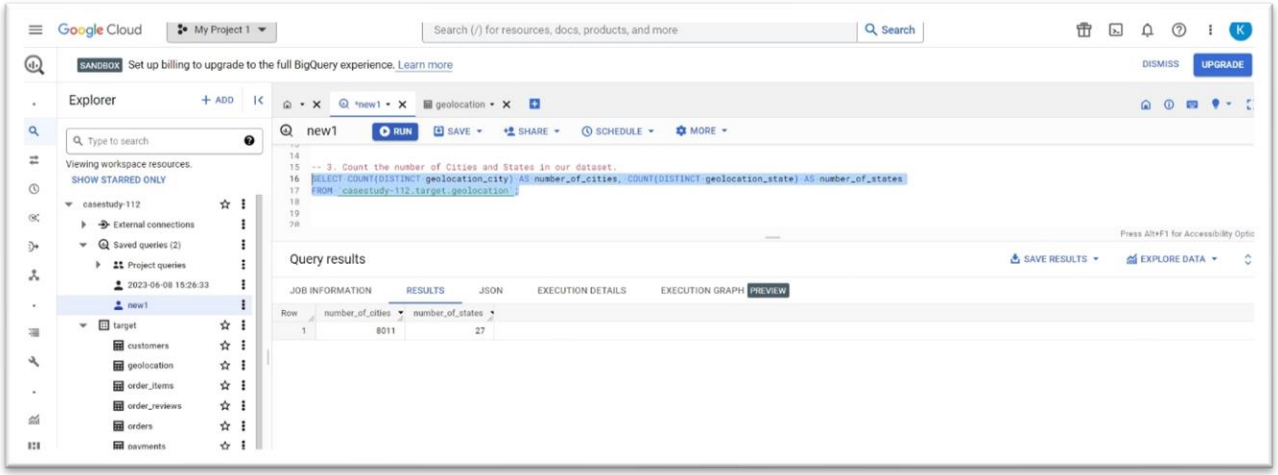


Figure 1.4: Counting the number of Cities and States in our dataset

Insights:

1. All columns in this example are saved as strings (VARCHAR), except customer_zip_code_prefix which is integer. This implies that the data might mostly be textual in nature, and that any values relating to dates or numbers are probably saved as strings.
2. The orders in this case were placed between 2 years, 25 months, or 773 days.
3. To analyze trends, seasonality, and overall order patterns over a certain time, it can be helpful to know the time range of the orders.
4. The dataset in this example has 27 distinct states and 8011 distinct cities. These figures can aid in our comprehension of the geographic distribution of our clientele or the scope of our dataset.

5. Analyzing the distribution of cities and states might reveal information about the diversity or concentration of our clientele in various geographic areas. It might be helpful for regional analysis, figuring out hotspots, or figuring out how far our company has spread across the nation or the globe.

Query 2:

```
#Q2: In-depth Exploration:

-- 1. Is there a growing trend in the no. of orders placed over the past years?
SELECT EXTRACT(YEAR FROM order_purchase_timestamp) AS year, COUNT(*) AS number_of_orders
FROM `casestudy-112.target.orders`
GROUP BY year
ORDER BY year;

-- 2. Can we see some kind of monthly seasonality in terms of the no. of orders being placed?
WITH cte AS
(
SELECT
  *,
  EXTRACT ( DATE FROM order_purchase_timestamp) AS order_date,
  EXTRACT ( YEAR FROM order_purchase_timestamp) AS order_year,
  EXTRACT ( MONTH FROM order_purchase_timestamp) AS order_month,
FROM `casestudy-112.target.orders`
)
SELECT
  order_month,
  order_year,
  COUNT(order_id) AS total_orders
FROM cte
GROUP BY
  order_month,
  order_year
ORDER BY
  order_year,
  order_month;

--3. During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)
--   0-6 hrs : Dawn
--   7-12 hrs : Mornings
--   13-18 hrs : Afternoon
--   19-23 hrs : Night
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 0 AND 6 THEN 'Dawn'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 7 AND 12 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 13 AND 18 THEN 'Afternoon'
    WHEN EXTRACT(HOUR FROM order_purchase_timestamp) BETWEEN 19 AND 23 THEN 'Night'
  END AS order_time_interval,
  COUNT(*) AS number_of_orders
FROM `casestudy-112.target.orders`
GROUP BY order_time_interval
ORDER BY number_of_orders DESC;
```

Figure 1.5: In-depth Exploration

Output screenshots:

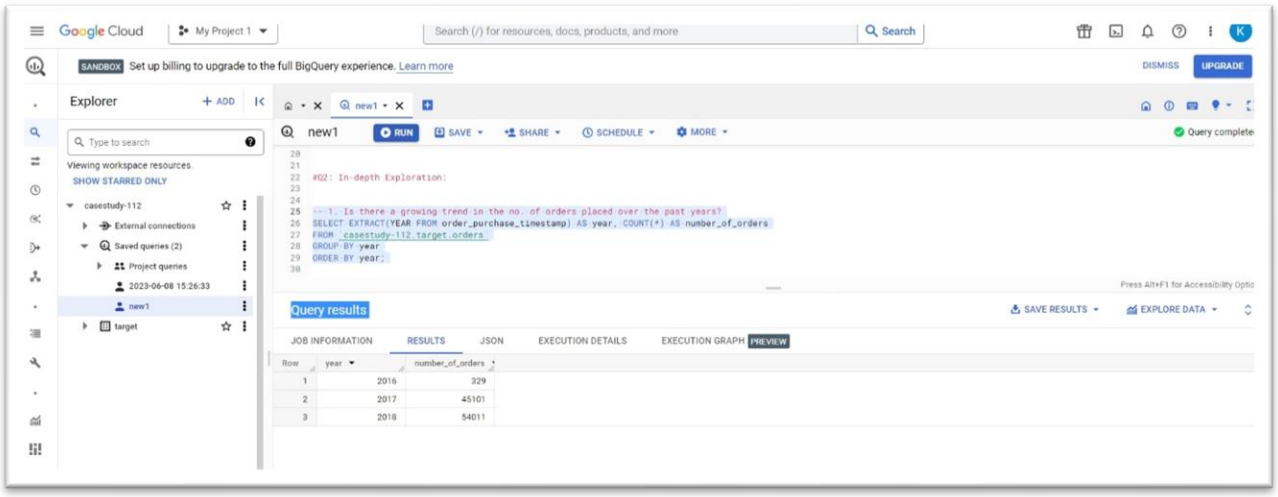


Figure 1.6: Is there a growing trend in the no. of orders placed over the past years?

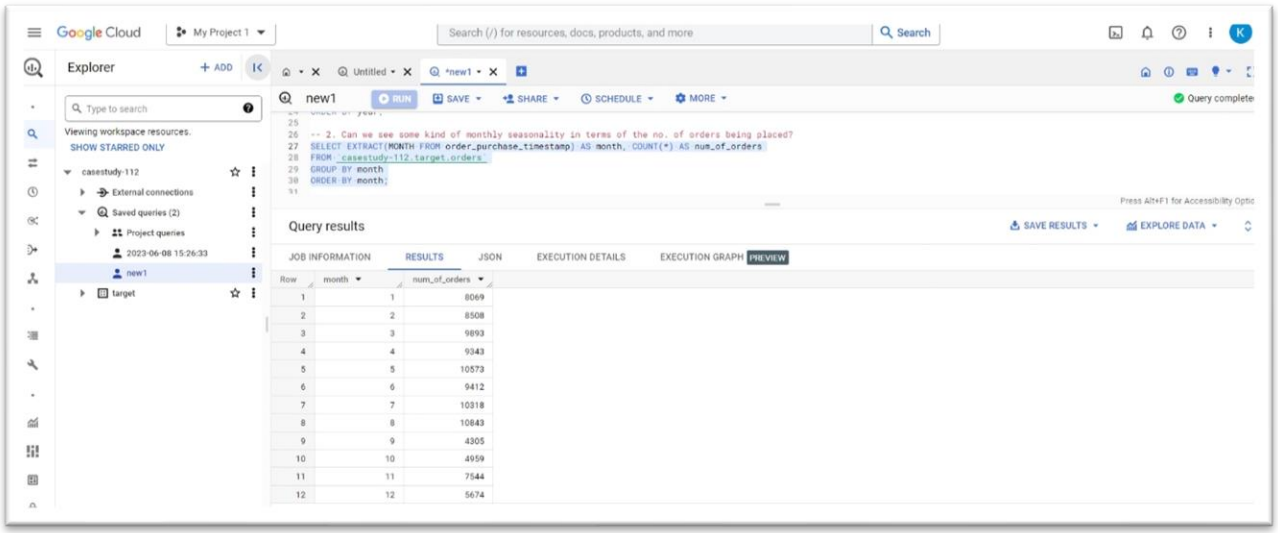


Figure 1.7: Can we see some kind of monthly seasonality in terms of the no. of orders placed?

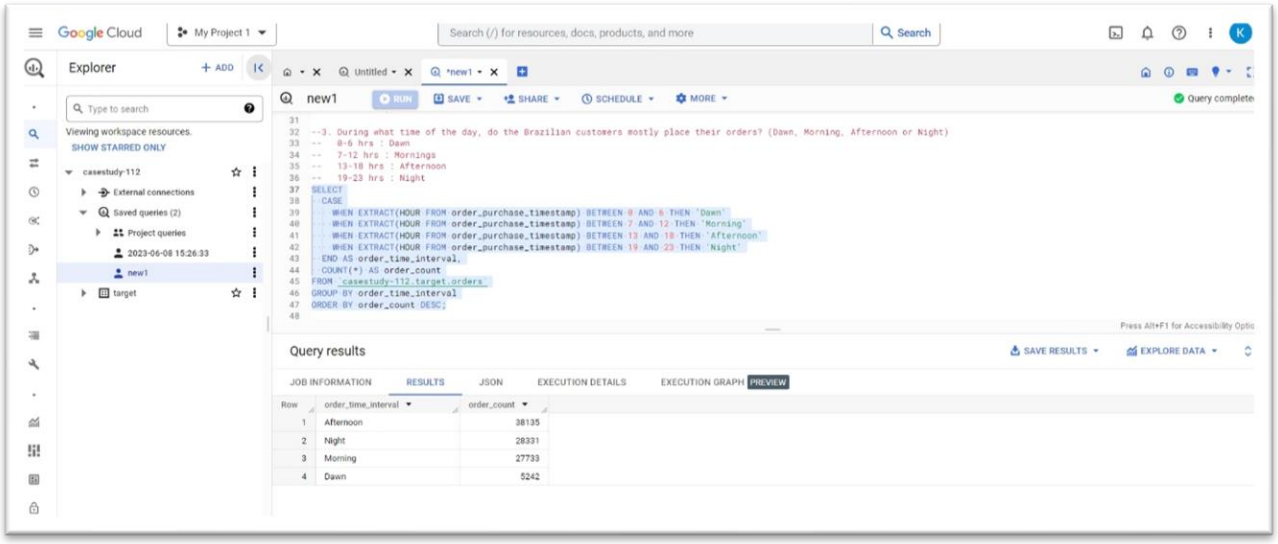


Figure 1.8: During what time of the day, do the Brazilian customers mostly place their orders? (Dawn, Morning, Afternoon or Night)

Insights:

1. There has been an upward trend in the number of orders over the past few years after examining the results. A favorable trend can be seen if the order number regularly rises year over year. If there are variations or a negative tendency, however, it points to a different pattern.
2. We see a seasonal trend for Nov 2017 where there was Black Friday and there is a huge increase in the orders placed.
3. There is also a growth trend in Jan 2017 and Jan 2018 where New Years is experienced, and people may have preordered for the Carnival in Feb
4. There is also an increase in orders in Q1 of 2018 during which FIFA World Cup was scheduled.
5. Understanding monthly seasonality can help with operational planning, marketing tactics, and consumer behavior. It can aid in better planning of promotional activities, inventory management optimization, and peak period identification and resource allocation.
6. Based on the hour component of the timestamp, the query divides the order timestamps into various time groups (Dawn, Morning, Afternoon, Night). The results are then sorted based on how many orders fall inside each time frame.
7. We can ascertain the time of day when Brazilian clients often place their orders by analyzing the data. We will learn more about their ordering habits and preferences. For instance, we discover that Brazilian clients frequently order more in the afternoons, indicating that this is a period when people want to shop online. By scheduling customer assistance personnel or launching focused marketing efforts during the busiest ordering periods, for example, we may operate more efficient operations with the aid of this information. Also, customers buy least during dawn.

Query 3:

```sql
#Q3. Evolution of E-commerce orders in the Brazil region:


-- 1. Get the month on month no. of orders placed in each state.
SELECT EXTRACT(MONTH FROM o.order_purchase_timestamp) AS order_month, c.customer_state,
COUNT(*) AS number_of_order
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY order_month, c.customer_state
ORDER BY order_month, c.customer_state;


-- 2. How are the customers distributed across all the states?
SELECT customer_state, COUNT(DISTINCT customer_id) AS total_custmers
FROM `casestudy-112.target.customers`
GROUP BY customer_state
ORDER BY total_custmers DESC;
```

Figure 1.9: Evolution of E-commerce orders in the Brazil region

Output screenshots:

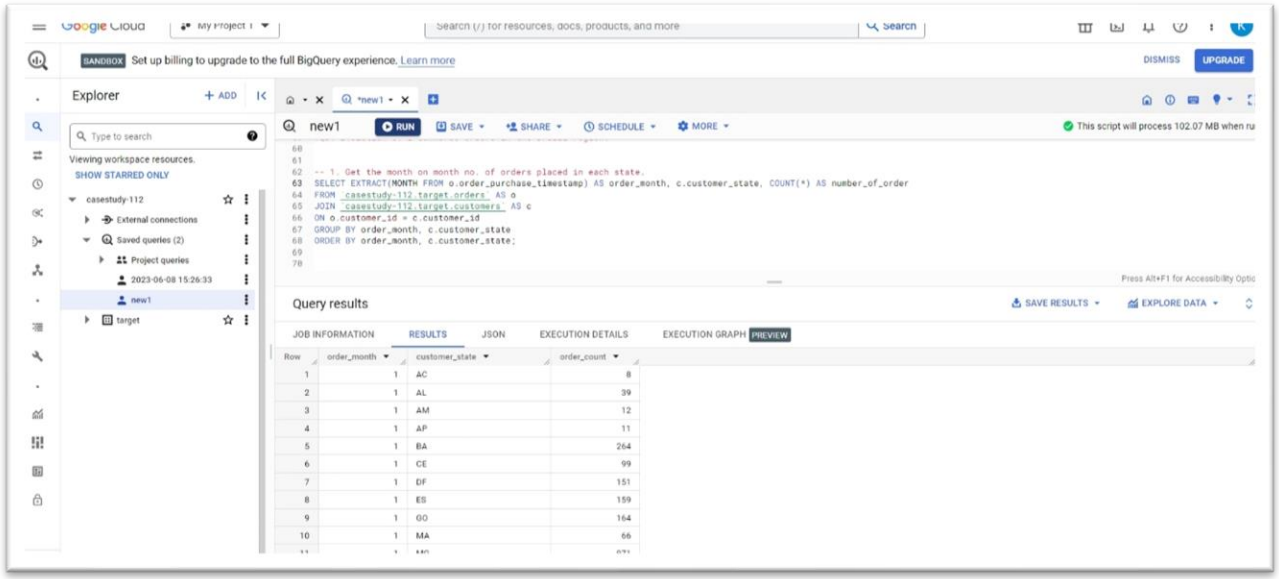


Figure 1.10: Get the month-on-month no. of orders placed in each state

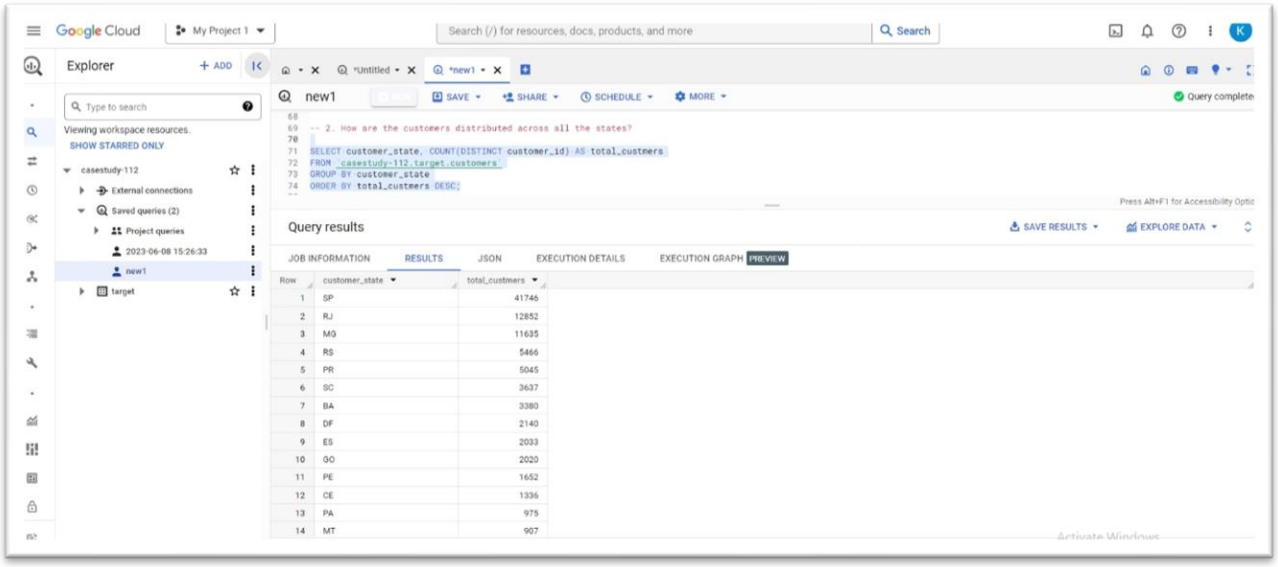


Figure 1.11: How are the customers distributed across all the states?

Insights:

1. We can learn more about the monthly order count for each state by examining the query's results. Over time, we can spot trends, patterns, or seasonality in the order volume for various states. We can use it to determine which states have consistently high order volumes and to pinpoint any months or states where order counts have significantly changed. Here in our data, we can see that every month the state called SP has the highest number of orders.
2. We can target marketing efforts in states with rising order volumes, spot potential operational issues in states with falling order volumes or optimize inventory management based on order trends across different states by analyzing
3. The distribution of clients across states will be shown by analyzing the query's results. Which states have the most customers, and which states have comparatively fewer consumers can be determined. Here the state called SP has the highest clients and the state called RR has the fewest clients. There are several uses for this information, including: Market targeting, Expansion opportunities and Customer service.
4. We can learn more about the geographic distribution of our client base, spot prospective growth areas, and make wise decisions to optimize our company strategy by looking at the customer distribution between states.

Query 4:

```sql
Q4: Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight and others.

--1. Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only). You can use the "payment_value" column in the payments table to get the cost of orders

SELECT
  ROUND((((total_payment_2018 - total_payment_2017) / total_payment_2017) * 100), 2) AS percentage_increase
FROM (
  SELECT
    SUM(CASE
      WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2017 AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value
      ELSE 0
    END) AS total_payment_2017,
    SUM(CASE
      WHEN EXTRACT(YEAR FROM o.order_purchase_timestamp) = 2018 AND EXTRACT(MONTH FROM o.order_purchase_timestamp) BETWEEN 1 AND 8 THEN p.payment_value
      ELSE 0
    END) AS total_payment_2018
  FROM
    `casestudy-112.target.payments` AS p
  JOIN
    `casestudy-112.target.orders` AS o
  ON
    p.order_id = o.order_id
);

--2. Calculate the Total & Average value of order price for each state.
SELECT customer_state,
        ROUND(SUM(p.payment_value),2) AS total_order_price,
        ROUND(AVG(p.payment_value),2) AS average_order_price
FROM `casestudy-112.target.payments` AS p
JOIN `casestudy-112.target.orders` AS o
ON p.order_id = o.order_id
JOIN `casestudy-112.target.customers` AS c
ON o.customer_id = c.customer_id
GROUP BY customer_state
ORDER BY total_order_price DESC;

--3. Calculate the Total & Average value of order freight for each state.
SELECT
  customer_state AS state,
  ROUND(SUM(oi.freight_value),2) AS total_freight,
  ROUND(AVG(oi.freight_value),2) AS average_freight
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.order_items` AS oi
ON o.order_id = oi.order_id
JOIN `casestudy-112.target.customers` AS c
ON c.customer_id = o.customer_id
GROUP BY state
ORDER BY total_freight DESC;
```

Figure 1.12: Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight, and others

Output screenshots:

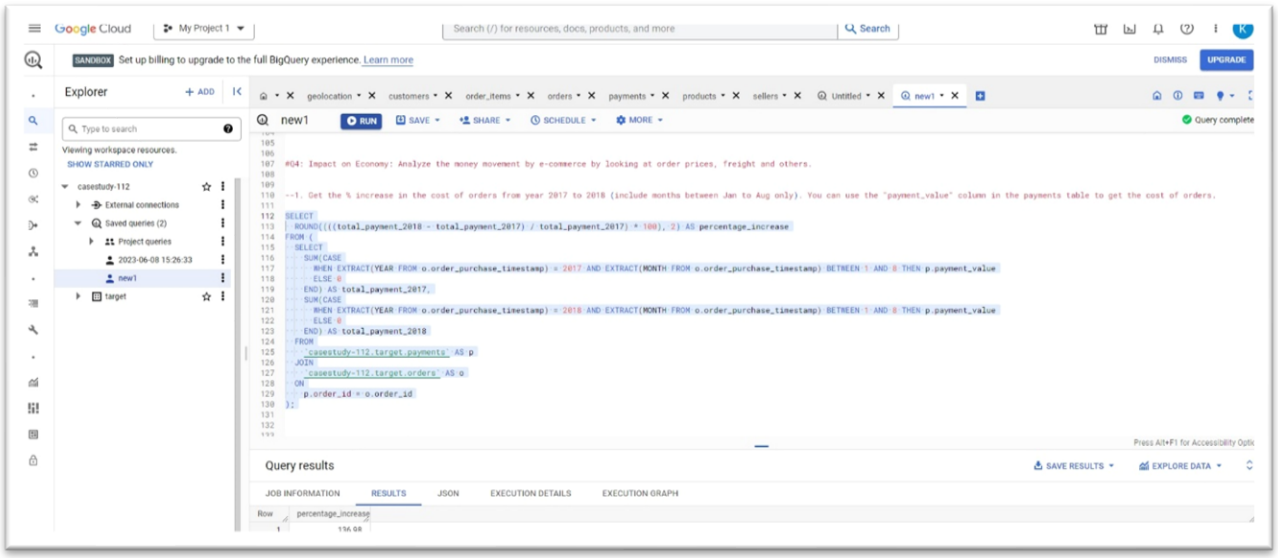


Figure 1.13: Get the % increase in the cost of orders from year 2017 to 2018 (include months between Jan to Aug only)

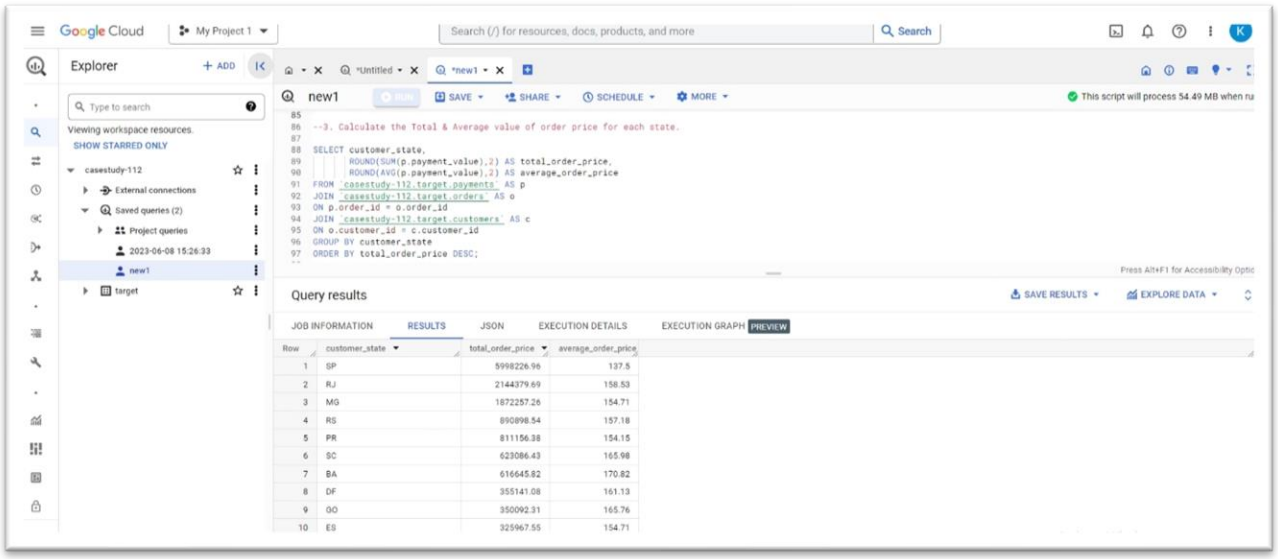


Figure 1.14: Calculate the Total & Average value of order price for each state

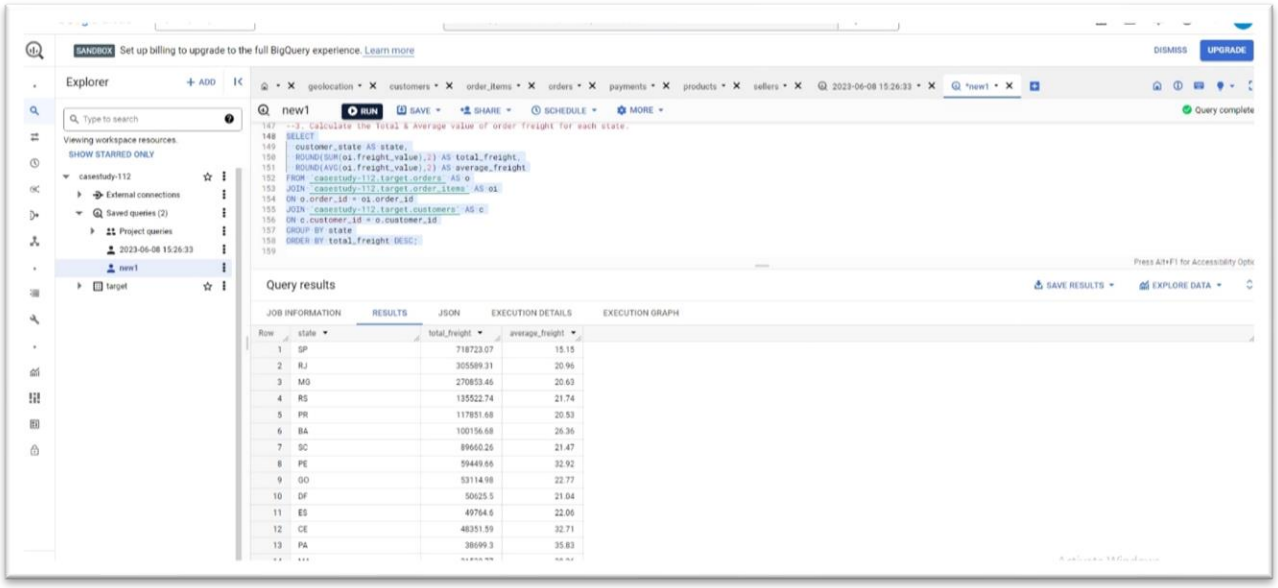


Figure 1.15: Calculate the Total & Average value of order freight for each state

Insights:

1. For both 2017 and 2018, only orders placed from January to August are considered.
2. To get the % increase, the query analyses the monthly prices between 2017 and 2018.
3. The findings tell us a growth rate of approximately 137% from 2017 to 2018.
4. The sum of all order prices for each state is displayed in the "total_order_price" column, which represents the total amount of orders placed.
5. The "average_order_price" column shows the normal order value for each state together with the average order price for that state.
6. We can find states with large total order values, which point to potentially profitable marketplaces, by analyzing the results.
7. To develop focused marketing or pricing strategies, it can be helpful to compare the average order prices across states to find areas with higher or lower average spending.
8. To obtain more understanding and make wise judgements based on the data, it's critical to consider the context of each state, such as population, economic variables, or customer behavior.
9. We can find states with high total freight costs, here in our case a state called SP, by analyzing the results, which could point to regions with higher shipping prices or logistical difficulties.
10. When optimizing logistics operations or pricing strategies, it might be helpful to discover regions with higher or lower average shipping prices by comparing the average order freight costs across states.
11. Understanding the differences in order freight rates between states can offer information about local shipping habits, supplier locations, or client preferences that can be used to optimize processes and cut costs.

Query 5:

```
#Q5. Analysis based on sales, freight and delivery time.

--1. Find the no. of days taken to deliver each order from the order's purchase date as delivery time. Also, calculate the difference (in days) between the estimated & actual delivery date of an order.
-- Do this in a single query. You can calculate the delivery time and the difference between the estimated & actual delivery date using the given formula: time_to_deliver = order_delivered_customer_date
-- - order_purchase_timestamp diff_estimated_delivery = order_estimated_delivery_date - order_delivered_customer_date
SELECT
  order_id,
  DATE_DIFF(DATE(order_delivered_customer_date), DATE(order_purchase_timestamp), DAY) AS delivery_time,
  DATE_DIFF(DATE(order_estimated_delivery_date), DATE(order_delivered_customer_date), DAY) AS diff_estimated_delivery
FROM
  `casestudy-112.target.orders`;

--2. Find out the top 5 states with the highest & lowest average freight value.
SELECT
  high.customer_state AS high_state,
  high.average_freight_value AS high_avg_freight,
  low.customer_state AS low_state,
  low.average_freight_value AS low_avg_freight
FROM
(
  SELECT
    c.customer_state,
    ROUND(AVG(p.freight_value),2) AS average_freight_value,
    ROW_NUMBER() OVER(ORDER BY (ROUND(AVG(p.freight_value),2))DESC) AS rowval1
  FROM `casestudy-112.target.orders` AS o
    JOIN `casestudy-112.target.order_items` AS p
    ON o.order_id = p.order_id
    JOIN `casestudy-112.target.customers` AS c
    ON o.customer_id = c.customer_id
  GROUP BY
    c.customer_state
  ORDER BY
    average_freight_value DESC
  LIMIT
    5
) AS high
JOIN
(
  SELECT
    c.customer_state,
    ROUND(AVG(p.freight_value),2) AS average_freight_value,
    ROW_NUMBER() OVER(ORDER BY (ROUND(AVG(p.freight_value),2))) AS rowval2
  FROM `casestudy-112.target.orders` AS o
    JOIN `casestudy-112.target.order_items` AS p
    ON o.order_id = p.order_id
    JOIN `casestudy-112.target.customers` AS c
    ON o.customer_id = c.customer_id
  GROUP BY
    c.customer_state
  ORDER BY
    average_freight_value
  LIMIT
    5
) AS low
ON high.rowval1 = low.rowval2;
```

Figure 1.16: Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight, and others

```sql
--3. Find out the top 5 states with the highest & lowest average delivery time.
WITH cte AS
(
  SELECT
    c.customer_state,
    ROUND(AVG(t1.delivery_time),2) AS avg_delivery_time
  FROM
  (
    SELECT
      *,
      TIMESTAMP_DIFF(order_delivered_customer_date, order_purchase_timestamp, day) AS delivery_time,
    FROM
      `casestudy-112.target.orders`
    WHERE
      order_status = 'delivered' AND order_delivered_customer_date IS NOT NULL
    ORDER BY
      order_purchase_timestamp
) AS t1
JOIN
  `casestudy-112.target.customers` AS c
  ON t1.customer_id = c.customer_id
GROUP BY
  c.customer_state
ORDER BY
  avg_delivery_time
)

SELECT
  c1.customer_state AS low_state,
  c1.avg_delivery_time AS low_avg_delivery_time,
  c2.customer_state AS high_state,
  c2.avg_delivery_time AS high_avg_delivery_time
FROM
(
  SELECT
    *,
    ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time DESC) AS rowval2
  FROM
    cte
  ORDER BY
    rowval2
) AS c2
JOIN
(
  SELECT
    *,
    ROW_NUMBER() OVER (ORDER BY cte.avg_delivery_time) AS rowval1
  FROM
    cte
  ORDER BY
    rowval1
) AS c1
ON
  c1.rowval1 = c2.rowval2
LIMIT
  5;
```

Figure 1.17: Impact on Economy: Find out the top 5 states with the highest & lowest average delivery time

```
--4. Find out the top 5 states where the order delivery is really fast as compared to the estimated date of delivery.
-- You can use the difference between the averages of actual & estimated delivery date to figure out how fast the delivery was for each state.
WITH delivery_speed AS (
  SELECT
    c.customer_state,
    AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY)) AS avg_delivery_speed,
    ROW_NUMBER() OVER (ORDER BY AVG(DATE_DIFF(o.order_delivered_customer_date, o.order_estimated_delivery_date, DAY))) AS rank_fastest
  FROM `casestudy-112.target.orders` AS o
  JOIN `casestudy-112.target.customers` AS c
  ON o.customer_id = c.customer_id
  WHERE o.order_delivered_customer_date IS NOT NULL AND o.order_estimated_delivery_date IS NOT NULL
  GROUP BY c.customer_state
)
SELECT customer_state, avg_delivery_speed
FROM delivery_speed
WHERE rank_fastest <= 5
ORDER BY avg_delivery_speed;
```

Figure 1.18: Impact on Economy: Analyze the money movement by e-commerce by looking at order prices, freight, and others

Insights:

1. Insights into the effectiveness of the delivery process, including any delays or early deliveries compared to the projected timeframe, can be gained by analyzing the delivery_time and diff_estimated_delivery columns.
2. These columns can be further examined to find trends, outliers, or elements that affect delivery times or discrepancies between estimated and actual delivery dates.
3. These insights can be applied to manage customer expectations, enhance customer satisfaction, optimize the delivery process, and improve logistics operations.
4. States with the highest average freight values like states called RR and PB may experience greater shipping prices due to reasons like remote locations, higher transportation costs, or supply chain difficulties.
5. It might be useful for our company to try to optimize logistics operations or save costs to locate places with relatively reduced shipping prices by looking at the states with the lowest average freight values like states such as SP and PR.
6. This data can help us develop focused initiatives, bargain for freight costs, or spot possible opportunities to reduce costs in our supply chain operations.
7. When assessing the data and drawing conclusions from these insights, it is crucial to consider additional elements like distance, transportation infrastructure, carrier availability, or regional economic variations.
8. Finding areas with effective delivery operations, quicker transit times, or solid logistics networks can be done by looking at the states like SP and PR with the lowest average delivery times and states called RR and AP with the highest average delivery times.

9. These insights can be helpful for our company looking to improve customer satisfaction, operational efficiency, delivery process optimization, and set reasonable expectations for customers based on regional delivery time patterns.
10. When evaluating the data and drawing conclusions from these insights, it's crucial to take additional elements into account, such as population density, the distinction between urban and rural locations, customer expectations, or unique logistical restrictions.
11. Using this information, our company can concentrate on areas where delivery efficiency improvements can be made, thereby improving customer experience and operational efficiencies.
12. Our company operating in these states called AC, RO, AP, and AM where average delivery speed is highest can take advantage of the quicker delivery times by highlighting their rapid and dependable service, thereby attracting more clients, and boosting client satisfaction.
13. These data can help us improve our operations, enhance customer experience, optimize logistics, or look for expansion prospects in areas with a track record of quick order delivery.

Query 6:

```
#Q6: Analysis based on the payments:


--1. Find the month on month no. of orders placed using different payment types.
SELECT
  FORMAT_TIMESTAMP('%Y-%m', o.order_purchase_timestamp) AS month,
  p.payment_type,
  COUNT(DISTINCT o.order_id) AS order_count
FROM `casestudy-112.target.orders` AS o
JOIN `casestudy-112.target.payments` AS p
ON o.order_id = p.order_id
GROUP BY month, p.payment_type
ORDER BY month;


--2. Find the no. of orders placed on the basis of the payment installments that have been paid.
SELECT payment_installments, COUNT(DISTINCT order_id) AS order_count
FROM `casestudy-112.target.payments`
GROUP BY payment_installments
ORDER BY payment_installments;
```

Figure 1.19: Analysis based on the payments

Output screenshots:

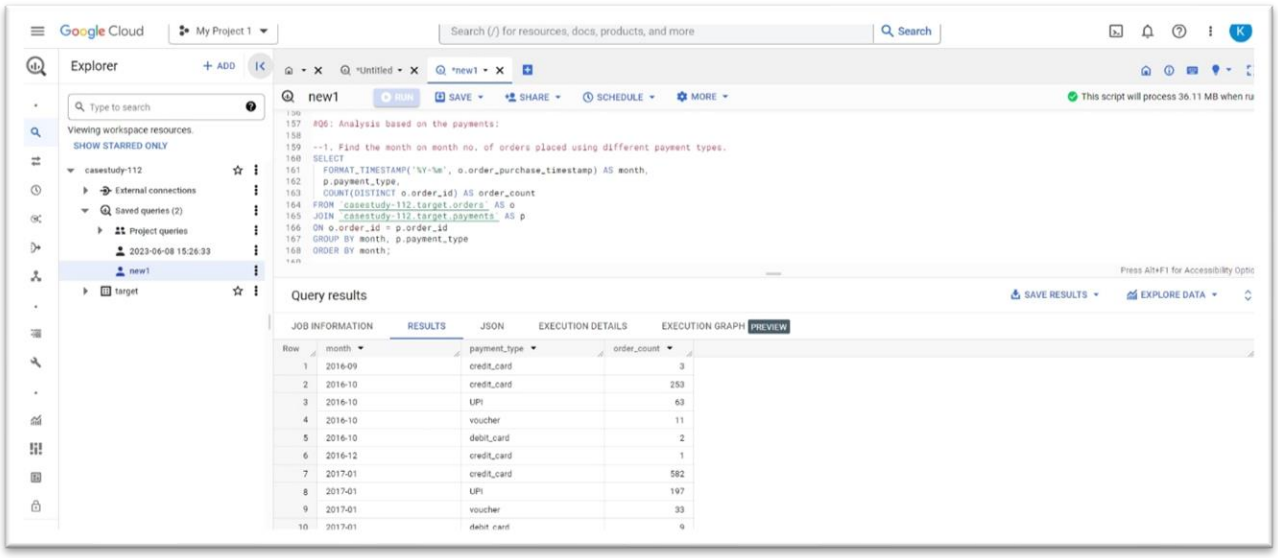


Figure 1.20: Find the month-on-month no. of orders placed using different payment types

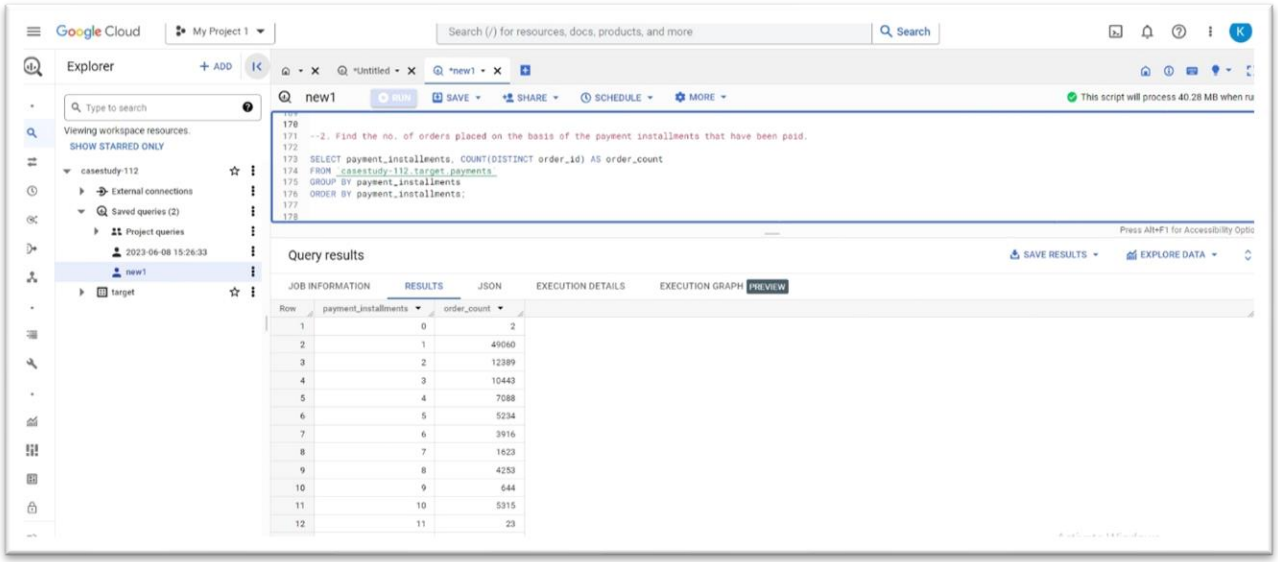


Figure 1.21: Find the no. of orders placed on the basis of the payment installments that have been paid

Insights:

1. We identify that credit card as a payment method was most used in November 2017.

2. To analyze seasonality, identify peak months, or evaluate the effects of marketing efforts or outside variables on consumer behavior, tracking the month-to-month trends in order counts can be helpful.
3. Based on the payment preferences noticed during various months, these insights might help firms optimize their payment procedures, customize marketing campaigns, or enhance customer experiences.
4. 49060 orders were placed where payment installment was 1.
5. This analysis can help determine whether payment installment alternatives are popular or preferred by clients.
6. Customers' preferences for budgeting or financing may be discerned by whether they tend to select a particular number of payment installments.
7. Monitoring the distribution of orders according to payment installments might reveal information about the buying habits of clients and their preference for flexible payment methods.

Insights:

1. Geographical dispersion:	The research reveals a wide geographic dispersion of clients, with the largest concentration in the state of São Paulo (SP), using data covering over 8,000 cities and 27 states. This can suggest a more expansive market or superior distribution systems in that area.
2. Order Trends and Seasonality:	Seasonality is evident, as seen by the significant increases in orders on Black Friday (November), New Year's Day (January), and major occasions such as the FIFA World Cup (Q1 2018). This suggests that taking advantage of customer purchasing behavior during these events might be achieved through marketing and operational planning.
3. Ordering Time of Day:	The afternoon (13–18 hours) is when most customers made their orders, with daybreak seeing the least amount of activity. According to Target Project SQL, afternoons are ideal

	for focused internet marketing and consumer interaction.
4. Freight expenses:	Due to logistical difficulties or distant locations, states like RR and PB have higher average shipping charges, whilst the state of SP has the highest overall freight expenses. Cost savings might result from shipping process optimization in these areas.
5. Customer Satisfaction:	According to Target Project SQL, the examination of customer feedback reveals high levels of satisfaction in some states. This suggests that high-scoring regions might serve as benchmarks for better service in other locations.
6. Product Popularity:	There seems to be a relationship between the quantity of product images and sales, suggesting that improved visual aids may influence buyer choice. Additionally, some product categories continuously exhibit stronger sales, which might influence inventory selections.

Table 1.1: Insights

Recommendations:

1. Targeted Marketing Campaigns:	Plan promotional activities, particularly around significant holidays and events, by utilizing data from seasonal surges in sales. For instance, increasing your investment in Black Friday and New Year's marketing may pay off.
2. Logistical Efficiency:	Consider streamlining the freight and delivery procedures, especially in places where shipping is expensive, or delivery periods are

	lengthy. This might involve modifying distribution networks or negotiating better terms with shipping partners.
3. Consumer Engagement:	To increase consumer engagement and generate more purchases during this peak hour, concentrate your marketing efforts on the afternoon time frame. This includes email campaigns, flash discounts, and promotions.

Table 1.2: Recommendations

Case Study Link:

https://github.com/MaharshiDSML/Target-Company-Case-Study-SQL

Chapter 2: Business Case Study 2- Data Exploration and Visualization

Problem Description:

1. One of the most popular media and video streaming platforms (OTT Platform) has over 10000 movies or tv shows available on their platform, as of mid-2021, they have over 222M Subscribers globally.

2. This tabular dataset consists of listings of all the movies and tv shows available on this platform, along with details such as - cast, directors, ratings, release year, duration, etc.

3. We need to analyze the data and generate insights that could help the company in deciding which type of shows/movies to produce and how they can grow the business in different countries.

4. The exploration would have a goal. As we explore the data, we will keep in mind that we want to answer which type of shows to produce and how to grow the business.

5. We will ensure that each recommendation is backed by data. The company is looking for data-driven insights, not personal opinions or anecdotes.

6. We will assume that we are presenting our findings to business executives who have only a basic understanding of data science. Therefore, we will avoid unnecessary technical jargon.

Business Questions to be answered from Analysis:

1. What type of content is available in different countries?
2. How has the number of movies released per year changed over the last 20-30 years?
3. Find the counts of each categorical variable both using graphical and nongraphical analysis.
4. Comparison of tv shows vs. movies.
5. What is the best time to launch a movie or a TV show?
6. Analysis of actors/directors of different types of shows/movies.
7. Does the platform have more focus on TV Shows than movies in recent years?
8. Understanding what content is available in different countries.

9. Which genre movies are more popular or produced more?
10. How many days will the movie be added to the platform after the release of the movie?

Methodology/Approach to Solve the Case Study:

1. Defining Problem Statement and Analyzing basic metrics.
2. Observations on the shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary.
3. Non-Graphical Analysis: Value counts and unique attributes.
4. Visual Analysis: Univariate, Bivariate after pre-processing of the data. Pre-processing involves unnesting the data in columns like Actor, Director, Country.
    a. For continuous variable(s): Distplot, Countplot, Histogram for univariate analysis.
    b. For categorical variable(s): Boxplot.
    c. For correlation: Heatmaps, Pairplots.
5. Missing Value & Outlier check.
6. Insights based on Non-Graphical and Visual Analysis.
7. Comments on the range of attributes.
8. Comments on the distribution of the variables and relationship between them.
9. Comments for each univariate and bivariate plot.
10. Business Insights: Should include patterns observed in the data along with what you can infer from it
11. Recommendations: Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand.

Analysis:

1. Data Preprocessing:

1. Dataset Shape: The dataset consists of **8807 rows** and **12 columns**. There are a few missing values in the "director," "cast," and "country" columns.
2. Data Types: 11 columns are strings (object), and 1 is an integer (release year).
3. Head of the DataFrame: Columns include show_id, type, title, director, cast, country, date_added, release_year, rating, duration, listed_in, description.
4. Checking basic statistical metrics: Of both numerical as well as categorical columns
5. Determining missing values: Of all numerical as well as categorical columns

Importing the libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the dataset

In [2]:
```
df = pd.read_csv("Netflix data.csv")
```

Checking the dataset

In [3]:
```
df.head(5)
```

Out[3]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

Figure 2.1: Importing libraries and loading the dataset

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Figure 2.2: Getting the basic information about the dataset

Getting the shape of data

Our data contains 8807 rows and 12 columns.

```
In [5]: df.shape
```

```
Out[5]: (8807, 12)
```

Getting statistical information on the integer column "release_year" of our data

```
In [6]: df.describe()
```

Out[6]:

	release_year
count	8807.000000
mean	2014.180198
std	8.819312
min	1925.000000
25%	2013.000000
50%	2017.000000
75%	2019.000000
max	2021.000000

Getting statistical information on 11 non-integer columns of our data

```
In [7]: df.describe(include = "object").T
```

Out[7]:

	count	unique	top	freq
show_id	8807	8807	s1	1
type	8807	2	Movie	6131
title	8807	8807	Dick Johnson Is Dead	1
director	6173	4528	Rajiv Chilaka	19
cast	7982	7692	David Attenborough	19
country	7976	748	United States	2818
date_added	8797	1767	January 1, 2020	109
rating	8803	17	TV-MA	3207
duration	8804	220	1 Season	1793
listed_in	8807	514	Dramas, International Movies	362
description	8807	8775	Paranormal activity at a lush, abandoned prope...	4

Figure 2.3: Getting statistical information about the dataset

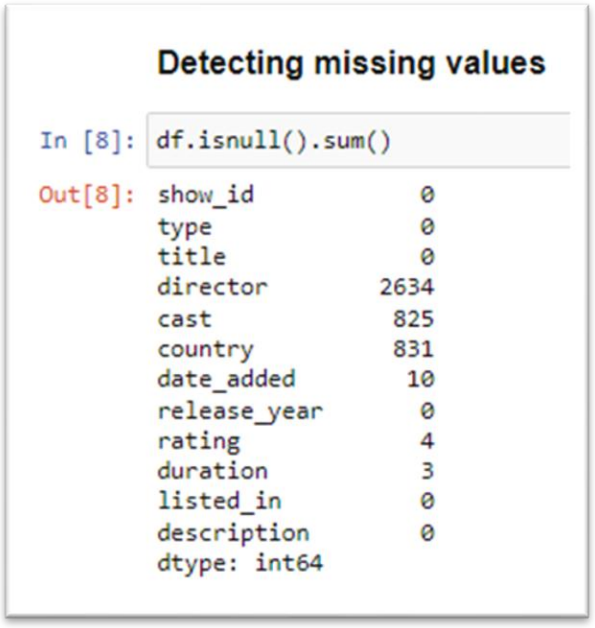


Figure 2.4: Detecting missing values

2. **Un-nesting the dataset:**
    a. Un-nesting the "director" column:
        i. **constraint = df['director'].apply(lambda x: str(x).split(', ')).tolist():** This line applies a lambda function to the "director" column of the DataFrame df. This lambda function takes each entry in the "director" column, converts it to a string (with str(x)), and then splits it by ', ' (a comma followed by a space) to create a list of director names. The .apply() function essentially applies this lambda function to each cell in the "director" column. The result is a list of lists where each inner list contains the director's names for a movie.
        ii. **df_director = pd.DataFrame(constraint, index=df['title']):** This line creates a new DataFrame called "df_director" using the constraint list created in the previous step as data. It sets the index of this new DataFrame to the "title" column of the original DataFrame df.
        iii. **df_director = df_director.stack():** This line "stacks" the DataFrame, essentially converting it from a multi-level DataFrame into a single-level Series. This operation makes it easier to manipulate the data.
        iv. **df_director = pd.DataFrame(df_director):** This line converts the Series obtained from the previous step back into a DataFrame.
        v. **df_director.reset_index(inplace = True):** This line resets the index of the DataFrame. It effectively removes the "title" column from the index and adds it as a regular column in the DataFrame. The inplace = True argument means that this operation modifies the DataFrame in place rather than creating a new DataFrame.

vi. **df_director = df_director[['title', 0]]:** This line selects only the "title" and 0-th column from the DataFrame. The 0-th column contains the director's names.

vii. **df_director.columns = ['title', 'director']:** This line renames the columns of the DataFrame to "title" and "director", providing more descriptive column names.

In summary, this code takes our original DataFrame with the "director" column, where directors are stored as comma-separated strings, and converts it into a new DataFrame with two columns: "title" and "director". Each row of this new DataFrame contains a single director for a particular movie, and any movie with multiple directors will have multiple rows in the resulting DataFrame.

```
In [9]: constraint = df['director'].apply(lambda x: str(x).split(', ')).tolist()
        df_director = pd.DataFrame(constraint, index = df['title'])
        df_director = df_director.stack()
        df_director = pd.DataFrame(df_director)
        df_director.reset_index(inplace = True)
        df_director = df_director[['title', 0]]
        df_director.columns = ['title','director']
        df_director
```

Out[9]:

	title	director
0	Dick Johnson Is Dead	Kirsten Johnson
1	Blood & Water	nan
2	Ganglands	Julien Leclercq
3	Jailbirds New Orleans	nan
4	Kota Factory	nan
...	...	...
9607	Zodiac	David Fincher
9608	Zombie Dumb	nan
9609	Zombieland	Ruben Fleischer
9610	Zoom	Peter Hewitt
9611	Zubaan	Mozez Singh

9612 rows × 2 columns

Figure 2.5: Un-nesting the "director" column

b. Similarly, we have un-nested the "cast", "country", "listed_in" columns

3. **Top 10 Popular Directors:**
    a. INSIGHT- We can see that "Rajiv Chilaka" is the most popular director on our platform with 22 movies that he has directed.
    b. Popularity of directors from the most prolific (at the top) to the least prolific (at the bottom) in terms of the number of unique titles they have directed.
    c. What does the following code do?
        i. **df_director.groupby(["director"])["title"].nunique():** This line groups the DataFrame df_director by the "director" column and calculates the number of unique titles associated with each director. The .groupby() method groups the data by director's names, and then the .nunique() function counts the number of unique titles for each director.
        ii. **result_director.sort_values(ascending = False):** After performing the grouping and counting, this line sorts the resulting Series, result_director, in descending order based on the number of unique titles. Directors with the highest number of unique titles will appear at the top of the Series due to the ascending = False parameter.
        iii. **result_director:** This is the Series that holds the count of unique titles for each director, sorted in descending order. The Series now shows directors in order of their prolificacy, with the most prolific directors appearing at the top.

```
In [13]: result_director = df_director.groupby(["director"])["title"].nunique()
         result_director = result_director.sort_values(ascending = False)
         result_director.head(10)

Out[13]: director
         nan                   2634
         Rajiv Chilaka           22
         Jan Suter               21
         Raúl Campos             19
         Suhas Kadav             16
         Marcus Raboy            16
         Jay Karas               15
         Cathy Garcia-Molina     13
         Martin Scorsese         12
         Youssef Chahine         12
         Name: title, dtype: int64
```

Figure 2.6: Top 10 Popular Directors

4. **Top 10 Popular Artists:**
    a. INSIGHT- We can see that "Anupam Kher" is the most popular artist on our platform with 43 movies that he has acted in.

```
In [14]: result_cast = df_cast.groupby(["cast"])["title"].nunique()
         result_cast = result_cast.sort_values(ascending = False)
         result_cast.head(10)

Out[14]: cast
         nan                 825
         Anupam Kher          43
         Shah Rukh Khan       35
         Julie Tejwani        33
         Naseeruddin Shah     32
         Takahiro Sakurai     32
         Rupa Bhimani         31
         Om Puri              30
         Akshay Kumar         30
         Yuki Kaji            29
         Name: title, dtype: int64
```

Figure 2.7: Top 10 Popular Artists

5. **Top 10 Content Consuming Countries:**
    a. INSIGHT- We can see that "United States" is the top consumer of our content where total 3689 number of content titles have been watched.

```
In [15]: result_country = df_country.groupby(["country"])["title"].nunique()
         result_country = result_country.sort_values(ascending = False)
         result_country.head(10)

Out[15]: country
         United States     3689
         India             1046
         nan                831
         United Kingdom     804
         Canada             445
         France             393
         Japan              318
         Spain              232
         South Korea        231
         Germany            226
         Name: title, dtype: int64
```

Figure 2.8: Top 10 Content Consuming Countries

6. **Top 10 Release Years:**
    a. INSIGHT- We can see that "2018" is the most common release_year for content on our platform with 1147 content titles having this release year.

```
In [16]: result_release_year = df.groupby(["release_year"])["title"].nunique()
         result_release_year = result_release_year.sort_values(ascending = False)
         result_release_year.head(10)

Out[16]: release_year
         2018    1147
         2017    1032
         2019    1030
         2020     953
         2016     902
         2021     592
         2015     560
         2014     352
         2013     288
         2012     237
         Name: title, dtype: int64
```

Figure 2.9: Top 10 Release Years

7. **Top 10 Ratings:**
    a. INSIGHT- We can see that "TV-MA" is the most common rated content on our platform with 3207 content titles having this rating.

```
In [17]: result_rating = df.groupby(["rating"])["title"].nunique()
         result_rating = result_rating.sort_values(ascending = False)
         result_rating.head(10)

Out[17]: rating
         TV-MA    3207
         TV-14    2160
         TV-PG     863
         R         799
         PG-13     490
         TV-Y7     334
         TV-Y      307
         PG        287
         TV-G      220
         NR         80
         Name: title, dtype: int64
```

Figure 2.10: Top 10 Ratings

8. **Top 10 Duration of Content:**
    a. INSIGHT- We can see that "1 season TV Shows" is the most common content on our platform with 1793 content titles.

```
In [18]: result_duration = df.groupby(["duration"])["title"].nunique()
         result_duration = result_duration.sort_values(ascending = False)
         result_duration.head(10)

Out[18]: duration
         1 Season     1793
         2 Seasons     425
         3 Seasons     199
         90 min        152
         97 min        146
         93 min        146
         94 min        146
         91 min        144
         95 min        137
         96 min        130
         Name: title, dtype: int64
```

Figure 2.11: Top 10 Duration of Content

9. **Top 10 Genres:**
    a. INSIGHT- We can see that "International Movies" is the most popular genre on our platform with a total of 2752 movies of this genre present.

```
In [19]: result_listed_in = df_listed_in.groupby(["listed_in"])["title"].nunique()
         result_listed_in = result_listed_in.sort_values(ascending = False)
         result_listed_in.head(10)

Out[19]: listed_in
         International Movies        2752
         Dramas                      2427
         Comedies                    1674
         International TV Shows      1351
         Documentaries                869
         Action & Adventure           859
         TV Dramas                    763
         Independent Movies           756
         Children & Family Movies     641
         Romantic Movies              616
         Name: title, dtype: int64
```

Figure 2.12: Top 10 Genres

10. A pie chart showing the type of content distribution:

a. INSIGHT- We can see that we have 69.6% Movies and 30.4% TV Shows, and so movies dominate the content type on our platform.

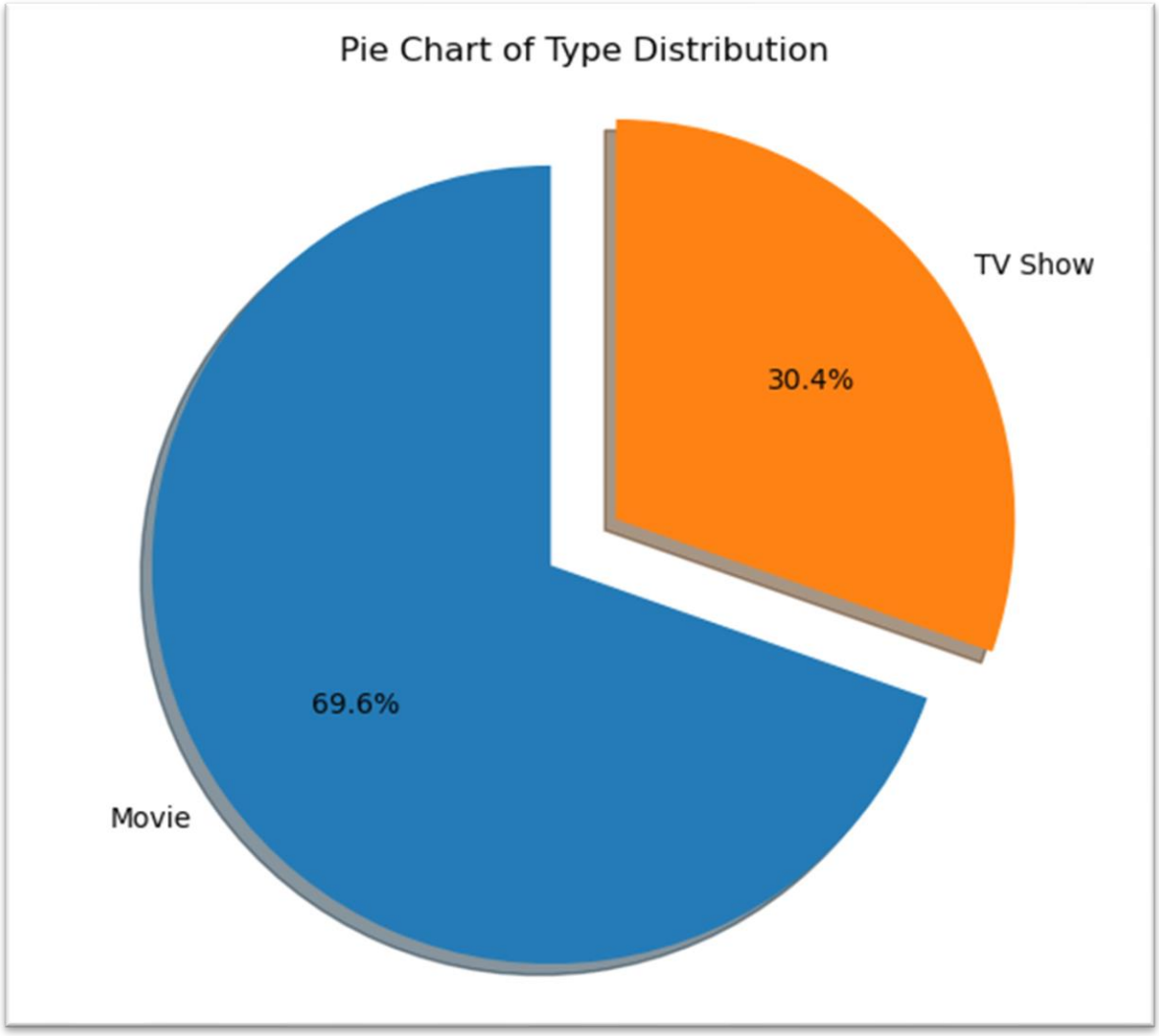


Figure 2.13: Pie chart showing content distribution

11. A bar chart showing top 10 directors by count:

a. INSIGHT- We can see that "Rajiv Chilaka" is the most popular director on our platform with 22 movies in which he has directed.

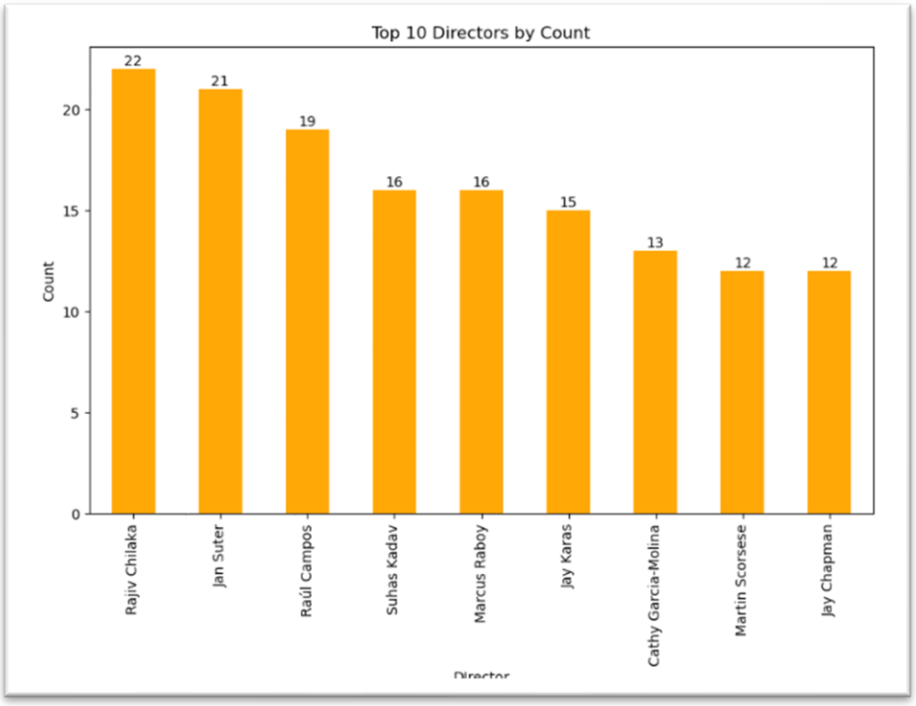


Figure 2.14: Bar chart of Top 10 Directors by Count

12. A bar chart showing top 10 directors by count:

a. INSIGHT- We can see that "Rajiv Chilaka" is the most popular director on our platform with 22 movies in which he has directed.

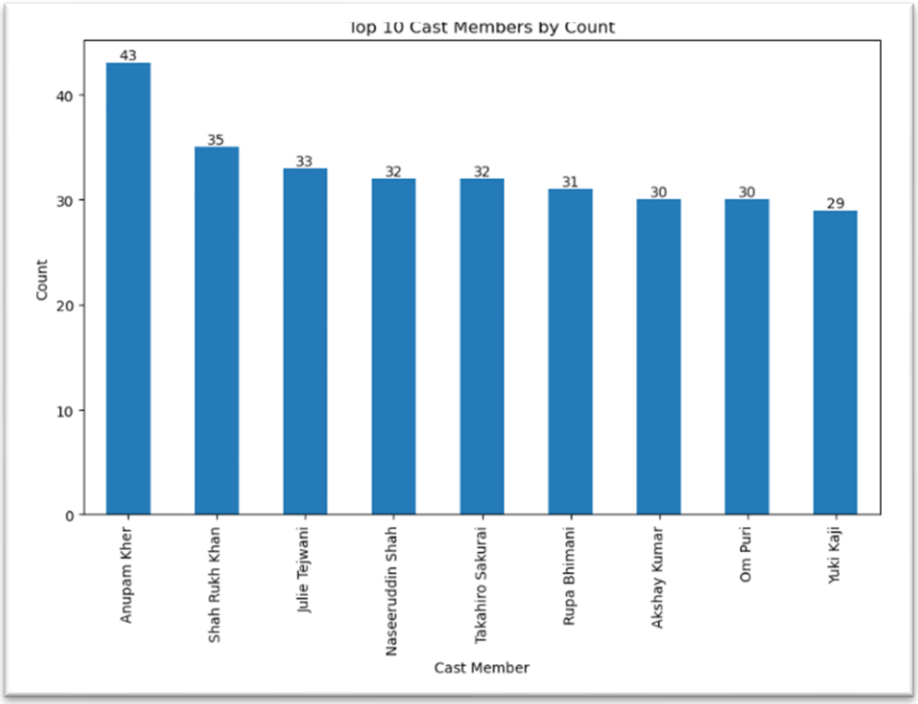


Figure 2.15: Bar chart of Top 10 Cast members by Count

13. A bar chart showing top 10 Countries by count:

a. INSIGHT- We can see that "United States" is the top consumer of our content where total 3689 number of content titles have been watched.

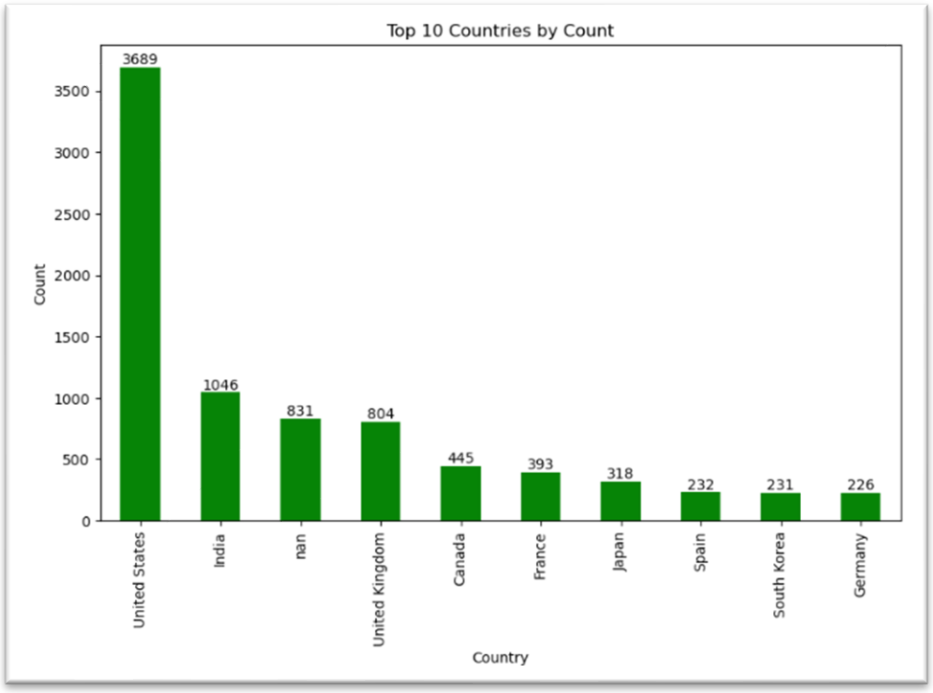


Figure 2.16: Bar chart of Top 10 Countries by Count

14. Analyzing the counts of "date_added" variable:

a. INSIGHT- We can see that "2020" is the top year in which most of our content got added to our platform with 109 number of content titles which have been added in that year.

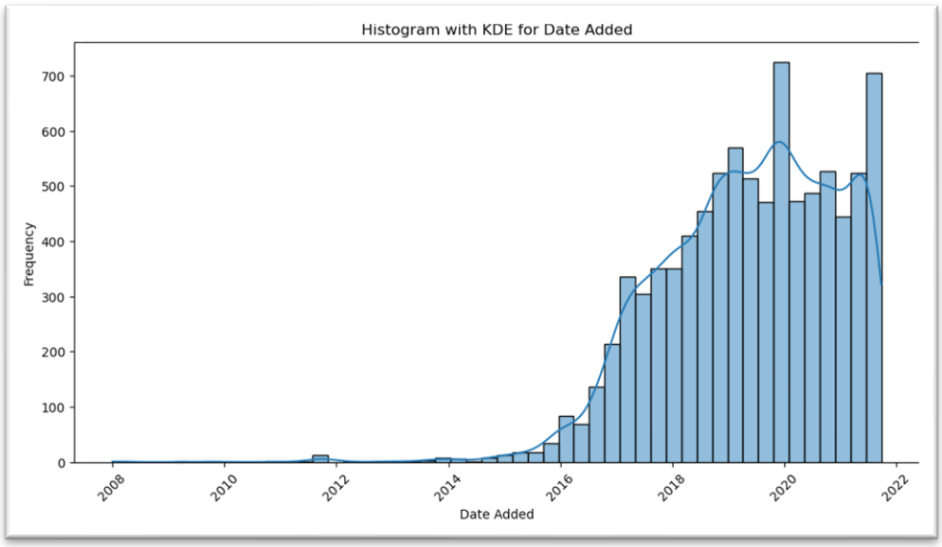


Figure 2.17: Histogram of "date_added" variable

15. Analyzing the counts of "release_year" numerical variable:

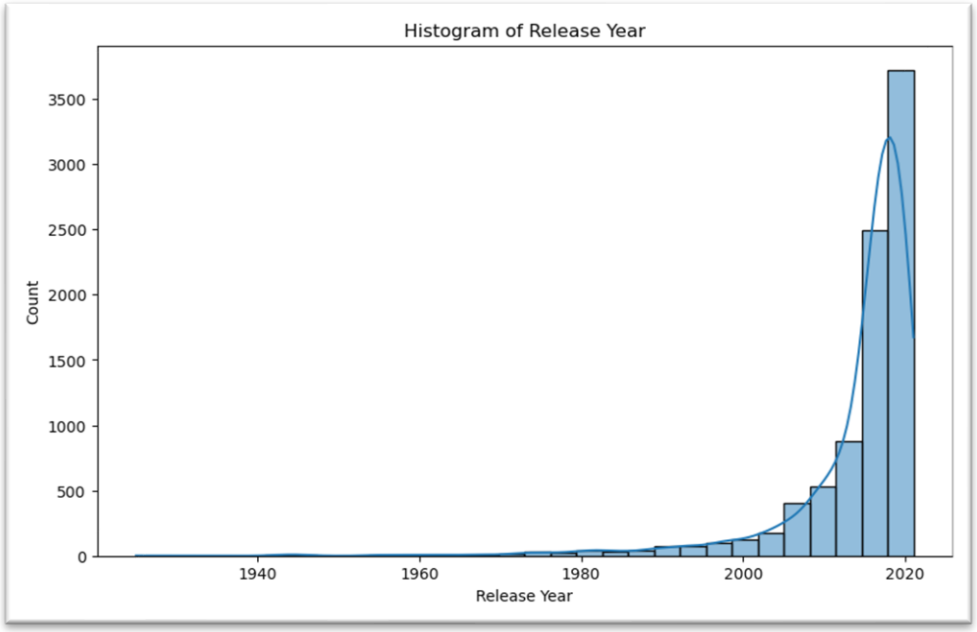


Figure 2.18: Histogram of "release_year" variable

16. Analyzing the counts of "rating" numerical variable using both graphical and nongraphical analysis:

```
In [33]: df["rating"].value_counts()

Out[33]: TV-MA       3207
         TV-14       2160
         TV-PG        863
         R            799
         PG-13        490
         TV-Y7        334
         TV-Y         307
         PG           287
         TV-G         220
         NR            80
         G             41
         TV-Y7-FV       6
         NC-17          3
         UR             3
         74 min         1
         84 min         1
         66 min         1
         Name: rating, dtype: int64
```

```
In [34]: ratings = df["rating"]

         # Create a count plot for ratings using Seaborn
         plt.figure(figsize=(10, 6))
         ax = sns.countplot(data=df, x=ratings, palette="Set2")  # Adjust the palette for color variations
         plt.xlabel("Rating")
         plt.ylabel("Count")
         plt.title("Count of Ratings")
         plt.xticks(rotation=45)  # Rotate the x-axis labels for better readability

         # Annotate the bars with their counts
         for p in ax.patches:
             ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()), ha='center', va='bottom')

         plt.show()
```

Figure 2.19: Code for counts of "rating" numerical variable using both graphical and nongraphical analysis

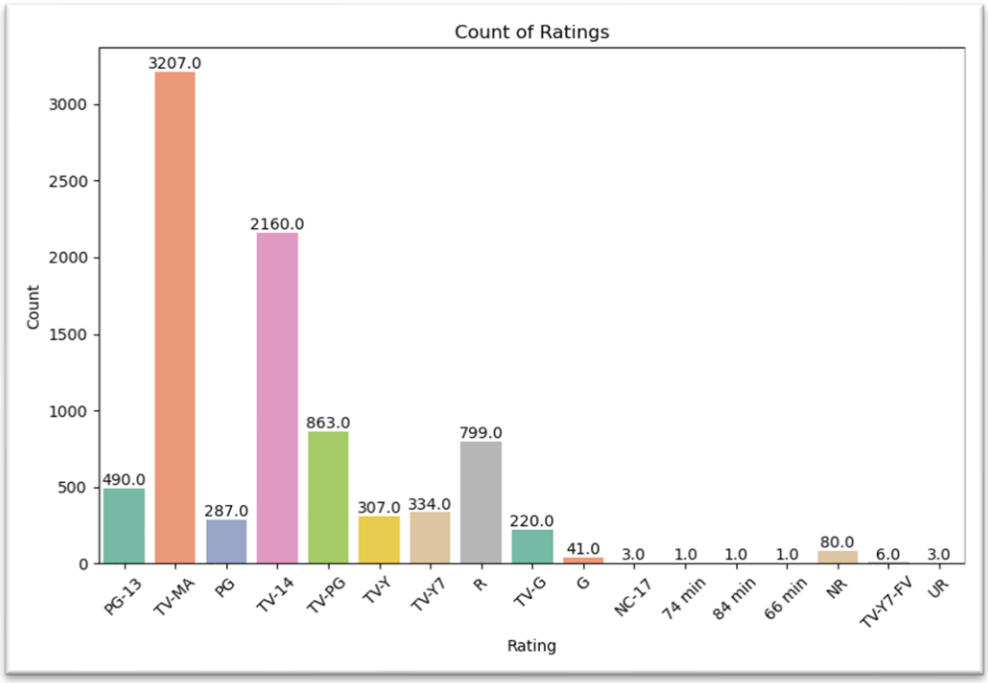


Figure 2.20: Bar chart of Count of Ratings

17. Analyzing the counts of "duration" numerical variable:

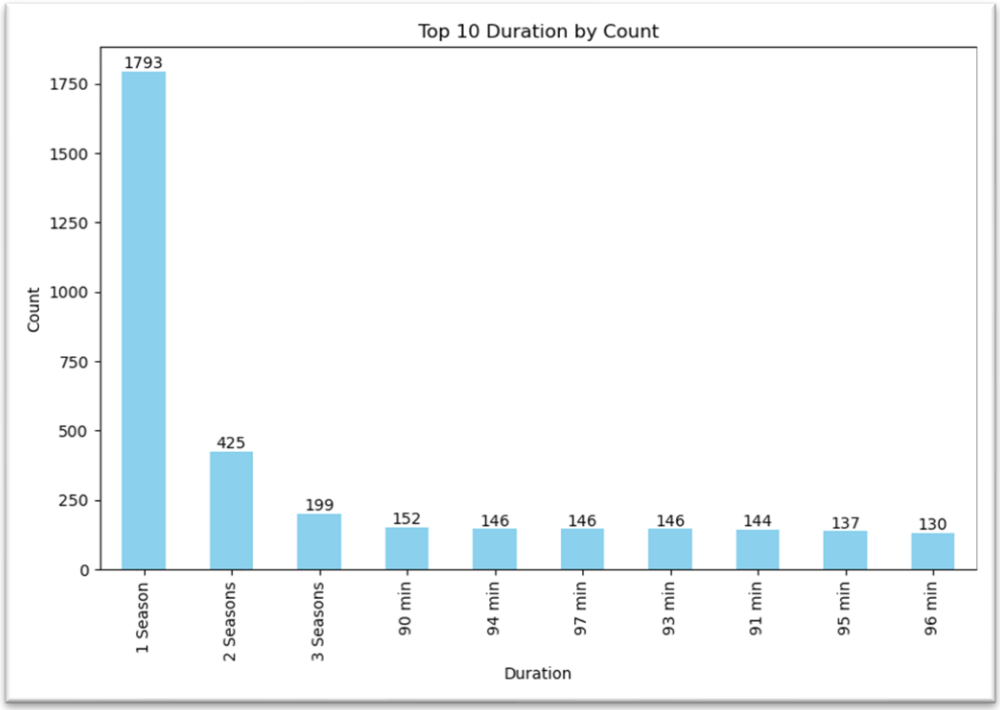


Figure 2.21: Bar chart showing top 10 “duration” by count

18. Analyzing the counts of "listed_in" categorical variable:

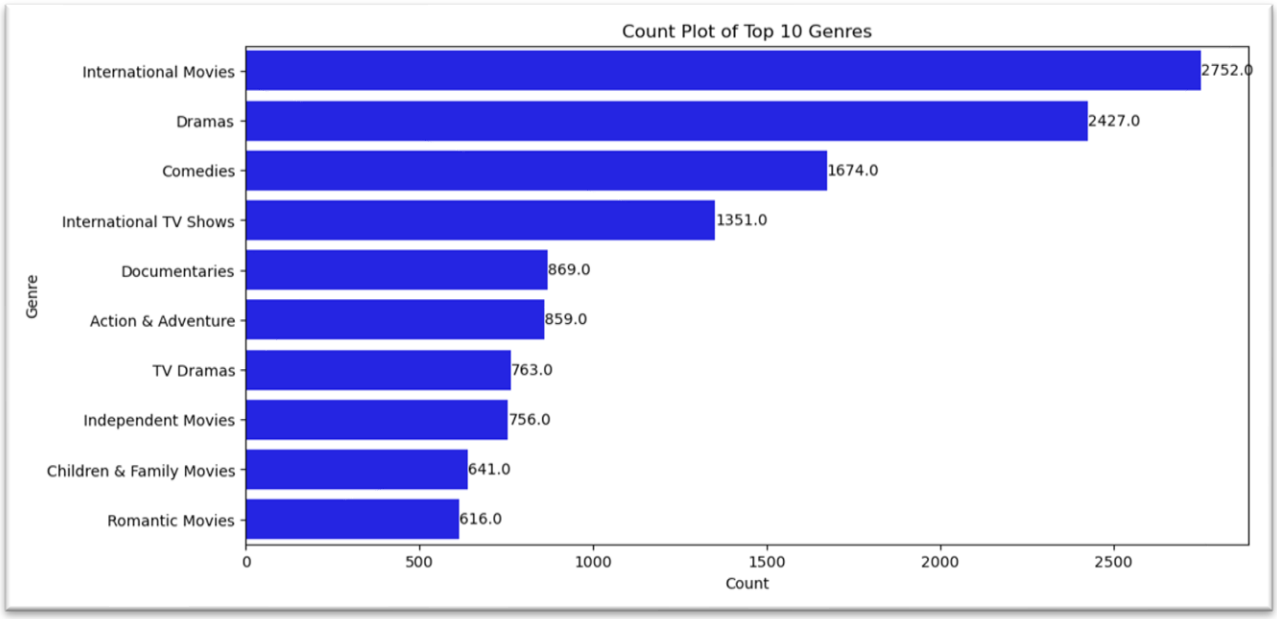


Figure 2.22: Count plot showing top 10 “genres” by count

19. Comparison of TV Shows vs. Movies:

a. Top 10 countries and the number of Movies produced in each country:

```
In [39]: # Filter for movies and group by country
         movies_data = df[df['type'] == 'Movie']
         movie_counts = movies_data['country'].value_counts().reset_index()
         movie_counts.columns = ['Country', 'Num_of_Movies']

         # Get the top 10 countries
         top_10_movies_countries = movie_counts.head(10)
         print(top_10_movies_countries)
```

	Country	Num_of_Movies
0	United States	2058
1	India	893
2	United Kingdom	206
3	Canada	122
4	Spain	97
5	Egypt	92
6	Nigeria	86
7	Indonesia	77
8	Turkey	76
9	Japan	76

Figure 2.23: Top 10 countries and the number of Movies produced in each country

```
In [41]: # Filter for TV shows and group by country
         tv_show_data = df[df['type'] == 'TV Show']
         tv_show_counts = tv_show_data['country'].value_counts().reset_index()
         tv_show_counts.columns = ['Country', 'Num_of_TV_Shows']

         # Get the top 10 countries
         top_10_tv_show_countries = tv_show_counts.head(10)
         print(top_10_tv_show_countries)
```

	Country	Num_of_TV_Shows
0	United States	760
1	United Kingdom	213
2	Japan	169
3	South Korea	158
4	India	79
5	Taiwan	68
6	Canada	59
7	France	49
8	Australia	48
9	Spain	48

Figure 2.24: Top 10 countries and the number of TV Shows produced in each country

20. Best Month to Launch a TV Show and a Movie

a. INSIGHT- We found that the best month to launch a TV Show on our platform was December.

b. INSIGHT- We found that the best month to launch a Movie on our platform was July.

```
In [50]: df['Date_added'] = pd.to_datetime(df['date_added'])

         # Extract the month number
         df['Month'] = df['Date_added'].dt.month

         # Separate TV shows and movies
         tv_shows = df[df['type'] == 'TV Show']
         movies = df[df['type'] == 'Movie']

         # Group by month and count the number of TV shows and movies
         tv_show_monthly_counts = tv_shows['Month'].value_counts().sort_index()
         movie_monthly_counts = movies['Month'].value_counts().sort_index()

         # Identify the month with the highest counts for TV shows and movies
         best_month_for_tv_shows = tv_show_monthly_counts.idxmax()
         best_month_for_movies = movie_monthly_counts.idxmax()

         # Get the count of tv shows in the best month
         highest_tv_shows_count = tv_show_monthly_counts.max()

         # Get the count of movies in the best month
         highest_movie_count = movie_monthly_counts.max()

         print("Best Month for TV Shows:", best_month_for_tv_shows)
         print("Best Month for Movies:", best_month_for_movies)
         print("Number of TV Shows Releases in the Best Month:", highest_tv_shows_count)
         print("Number of Movie Releases in the Best Month:", highest_movie_count)

         Best Month for TV Shows: 12.0
         Best Month for Movies: 7.0
         Number of TV Shows Releases in the Best Month: 266
         Number of Movie Releases in the Best Month: 565
```

Figure 2.25: Best Month to Launch a TV Show and a Movie

Insights:

1. **Release Year Trends:**	Over the years, the number of movies and TV shows added to the platform has seen a substantial increase. This indicates a growing library content and a significant investment in the platform's offerings. We can see that in the year 2020 most of our content was added on our platform.
2. **Content Types:**	The dataset includes both movies and TV shows, and it's clear that movies dominate in terms of the number of titles available. This

	could suggest that movies are more popular or are easier to produce than TV shows. We can see there are 6131 movies and 2676 TV Shows on our platform, and in terms of percentages, we have 69.6% Movies and 30.4% TV Shows. This has been demonstrated with non-graphical analysis and creating visualization on cell numbers.
3. **Consumption of content country-wise:**	There is a diverse set of countries listed as consumers for content on our platform. This demonstrates the company's commitment to offering content from around the world and catering to a global audience. We have identified top 10 Content Consuming Countries, and we can see that "United States" is the top consumer of our content with 3689 titles consumed in the country.
4. **Content Duration:**	While movies have specific durations in minutes, TV shows often have a number of seasons listed. This highlights the different formats of content available on the OTT platform. We have identified the top 10 durations of our content, and we can see that "1 season TV Shows" is the most common content on our platform.
5. **Ratings:**	The dataset includes TV ratings, which can help users make informed decisions about the content they want to watch. We have identified the top 10 Ratings of content, and we can see that "TV-MA" is the most common rating given to content on our platform with 3207 titles given this rating.
6. **Top 10 Directors:**	We have identified the directors who have directed most movies and TV shows on our platform. This can help you understand which directors are in high demand and may have a strong fan following.
7. **Top 10 Countries:**	We have identified Top 10 Content Consuming Countries, and we can see that "United States" is the top consumer of our

	content with 3689 titles consumed in the country.
8. **Top 10 Release Years:**	We have identified release year trends and determined which release years are the most represented in the top 10. Are there certain years that saw a significant number of releases? YES -- We have identified that in the year 2020 most of our content was added to our platform.
9. **Top 10 Artists (Cast):**	We have identified the actors who have appeared in most movies and TV shows on our platform. This can help you understand which actors are in high demand and may have a strong fan following.
10. **Top 10 Ratings (Cast):**	We have identified Top 10 ratings of our content, and we can see that "1 season TV Shows" is the most common content on our platform.
11. **Top 10 Genres:**	We have identified the most frequently occurring genres among the top 10. This can help you understand which genres are preferred by viewers.
12. **Best Week and Month to Launch a TV Show and a Movie:**	We found that the best week to launch a TV Show on our platform was the 27th week. We found that the best week to launch a Movie on our platform was the 1st week. We found that the best month to launch a Movie on our platform was July.
13. **Missing values detection:**	We have identified missing values in our data and have employed techniques to handle this missing data. Before pre-processing data, we have the missing values data which is clearly shown non-graphically in figure 2.4.

Table 2.1: Insights

Recommendations:

1. **Content Diversity:**	Netflix should continue to produce a diverse range of content, including both movies and TV shows, as the data shows that both types have their audience. This diversification can help cater to a broader viewer base.
2. **Focus on Movies:**	While movies dominate the platform, they continue to invest in high-quality movies to maintain and expand the movie library. We should consider user preferences and genres that have been successful in the past.
3. **Global Expansion:**	The analysis reveals that Netflix offers content from various countries and has a global presence. Netflix should continue to invest in producing content from different countries to cater to a global audience. Localized content can attract more subscribers in specific regions.
4. **Understanding Viewer Preferences:**	We analyzed viewer ratings and preferences for different genres (as indicated by the "Listed_in" column). We have identified Top 10 Ratings of content, and we can see that "TV-MA" is the most common rating given to content on our platform with 3207 titles given this rating. This information can guide the production of content that aligns with viewer interests.
5. **Timing Matters:**	The data can be used to determine the best time to launch TV shows. Analyzing viewer behavior and preferences by release year can help Netflix optimize their content release schedules. We found that the best week to launch a TV Show on our platform was 27th week and the best week to launch a movie was 1st week. We also found that the best month to launch a TV Show on our platform was December and the best month to launch a movie was July.
6. **User Feedback:**	We can implement mechanisms for users to provide feedback and suggestions on the platform. Analyzing user comments and reviews can help in tailoring content to user expectations.
7. **Content Duration:**	We can ensure that TV shows and movies both are produced equally and are tagged

	correctly with their respective durations. Clear and consistent information helps users in making selections.
8. **Recognizing Successful Actors/Directors:**	We have identified actors and directors who have consistently been part of successful content, which we have given in insights section. We should collaborate with these professionals more frequently as they have a track record of attracting viewers.
9. **Tailored Marketing:**	We can use data to create personalized content recommendations for individual users. This can increase engagement and retention on our platform.
10. **A/B Testing:**	We must conduct A/B testing of content to understand which types of shows or movies perform better. This can help refine content strategies and optimize investments.
11. **Balanced Investment:**	We can maintain a balanced investment in both original content and licensed content. Licensing popular titles can attract new subscribers while original content can retain existing ones.
12. **Viewer Retention:**	We need to pay attention to data on viewer churn rates. Understand why subscribers leave the platform and take steps to improve retention, such as producing content that appeals to a wider audience.
13. **Data-Driven Decision-Making:**	We encourage data-driven decisions in the production and content acquisition process. We must regularly analyze user data to understand changing viewer preferences and trends.

Table 2.2: Recommendations

Case Study Link:

https://github.com/MaharshiDSML/Netflix-Company-Case-Study-Python-Libraries

Chapter 3: Business Case Study 3- Descriptive Statistics and Probability

Problem Description:

1. A leading brand in the field of fitness equipment provides a product range including machines such as treadmills, exercise bikes, gym equipment, and fitness accessories to cater to the needs of all categories of people.

2. The market research team of the company wants to identify the characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills for the new customers.

3. The team decides to investigate whether there are differences across the product with respect to customer characteristics.

4. Product Portfolio:
    a. The KP281 is an entry-level treadmill that sells for $1,500.
    b. The KP481 is for mid-level runners that sell for $1,750.
    c. The KP781 treadmill has advanced features that sell for $2,500.

Business Questions to be answered from Analysis:

1. Check if features like marital status and age have any effect on the product purchased (using Countplot, histplots, boxplots etc.).
2. Representing the marginal probability like - what percentage of customers have purchased KP281, KP481, or KP781 in a table? *(we can use pandas.crosstab here)*
3. Check correlation among different factors using heat maps or pair plots.
4. With all the above steps you can answer questions like: What is the probability of a male customer buying a KP781 treadmill?
5. Customer Profiling - Categorization of users.
6. Probability- marginal, conditional probability.
7. Some recommendations and actionable insights, based on the inferences.

Methodology/Approach to Solve the Case Study:

1. We will perform descriptive analytics to create a customer profile for each treadmill by developing appropriate tables and charts.
2. For each treadmill, we will construct two-way contingency tables and compute all conditional and marginal probabilities along with their insights/impact on the business.
3. We will import the dataset and do usual data analysis steps like checking the structure & characteristics of the dataset.
4. We will detect Outliers (using boxplot, “describe” method by checking the difference between mean and median).
5. We will define Problem Statement and Analyze basic metrics:
    a. Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary.
12. We will perform Non-Graphical Analysis: Value counts and unique attributes.
13. We will perform Visual Analysis - Univariate & Bivariate:
    a. For continuous variable(s): Distplot, Countplot, Histogram for univariate analysis.
    b. For categorical variable(s): Boxplot.
    c. For correlation: Heatmaps, Pairplots.
14. We will perform Missing Value & Outlier Detection.
15. Business Insights based on Non-Graphical and Visual Analysis:
    a. Comments on the range of attributes.
    b. Comments on the distribution of the variables and relationship between them.
    c. Comments for each univariate and bivariate plot.
16. Recommendations: Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand

Analysis:

1. Analyzing basic metrics:

1. The shape of the data is (180, 9).
2. Data types of all the attributes - There are 6 integer and 3 string attributes in the dataset.
3. Name of attributes - Age, Gender, Education, Marital Status, Usage, Fitness, Income and Miles.
4. There are 180 rows in the data.
5. No null or missing values present in any row or column.

2.1 Checking the shape of the dataset

```
In [4]: df.shape
Out[4]: (180, 9)
```

2.2 Checking the structure and characteristics of the dataset

```
In [5]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    object
 1   Age            180 non-null    int64
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

Figure 3.1: Analyzing basic metrics (1)

2.3 Summary statistics

```
In [6]: df.describe()
```

Out[6]:

	Age	Education	Usage	Fitness	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.194444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.863605
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.000000
25%	24.000000	14.000000	3.000000	3.000000	44058.750000	66.000000
50%	26.000000	16.000000	3.000000	3.000000	50596.500000	94.000000
75%	33.000000	16.000000	4.000000	4.000000	58668.000000	114.750000
max	50.000000	21.000000	7.000000	5.000000	104581.000000	360.000000

```
In [7]: df.describe(include = "object")
```

Out[7]:

	Product	Gender	MaritalStatus
count	180	180	180
unique	3	2	2
top	KP281	Male	Partnered
freq	80	104	107

Figure 3.2: Analyzing basic metrics (2)

2. **Product Distribution:**

   Using pandas.crosstab to determine the count of each treadmill product (KP281, KP481, KP781) in the dataset and calculating the percentage distribution of each product to understand the popularity.

   1. INSIGHT: Top selling product is KP281 with 80 units sold which comprises 44.44% of total sales. The least selling product is KP781 with 40 units sold which comprises 22.22% of total sales.

```
In [8]: product_distribution = pd.crosstab(index=df['Product'], columns='count')
        product_percentage_distribution = (product_distribution / len(df)) * 100

        print("Product Distribution:")
        print(product_distribution)
        print("\nPercentage Distribution:")
        print(product_percentage_distribution)
```

```
Product Distribution:
col_0    count
Product
KP281       80
KP481       60
KP781       40

Percentage Distribution:
col_0        count
Product
KP281    44.444444
KP481    33.333333
KP781    22.222222
```

Figure 3.3: Product Distribution

3. Average Age, Income, and Usage:

Computing basic metrics such as mean and median for Age, Income, and Usage to understand the central tendency.

```
In [9]: average_age = df['Age'].mean()
        median_age = df['Age'].median()

        average_income = df['Income'].mean()
        median_income = df['Income'].median()

        average_usage = df['Usage'].mean()
        median_usage = df['Usage'].median()

        print(f"Average Age: {average_age:.2f}, Median Age: {median_age}")
        print(f"Average Income: ${average_income:,.2f}, Median Income: ${median_income:,.2f}")
        print(f"Average Usage: {average_usage:.2f}, Median Usage: {median_usage}")
```

```
Average Age: 28.79, Median Age: 26.0
Average Income: $53,719.58, Median Income: $50,596.50
Average Usage: 3.46, Median Usage: 3.0
```

Figure 3.4: Average Age, Income, and Usage

4. **Gender Distribution:**

   Analyzing the distribution of gender to understand the gender representation in the dataset.

   1. INSIGHT: 57.78% of our total customers are males and 42.22% are females.

```
In [10]: gender_distribution = df['Gender'].value_counts()
         gender_percentage_distribution = (gender_distribution / len(df)) * 100

         print("Gender Distribution:")
         print(gender_distribution)
         print("\nPercentage Distribution:")
         print(gender_percentage_distribution)

         Gender Distribution:
         Male      104
         Female     76
         Name: Gender, dtype: int64

         Percentage Distribution:
         Male      57.777778
         Female    42.222222
         Name: Gender, dtype: float64
```

Figure 3.5: Gender Distribution

5. **Education Level Distribution:**

   Exploring the distribution of education levels to identify the educational background of customers.

   1. INSIGHT: Majority of our customers are having moderate level of education. Analyzing the distribution of gender to understand the gender representation in the dataset.

```
In [11]: education_distribution = df['Education'].value_counts()
         education_percentage_distribution = (education_distribution / len(df)) * 100

         print("Education Distribution:")
         print(education_distribution)
         print("\nPercentage Distribution:")
         print(education_percentage_distribution)
```

```
Education Distribution:
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: Education, dtype: int64

Percentage Distribution:
16    47.222222
14    30.555556
18    12.777778
15     2.777778
13     2.777778
12     1.666667
21     1.666667
20     0.555556
Name: Education, dtype: float64
```

Figure 3.6: Education Level Distribution

6. **Non-Graphical Analysis:** Value counts and unique attributes

3.1 Value counts and unique attributes of Product column

Insight: Top selling product is KP281 with sales of 80 units, whereas 60 units of KP481 and 40 units of KP781 were sold.

```
In [12]: print("Total count of Product:")
         print(df["Product"].value_counts())

         print("\nUnique attributes of Product:")
         print(df["Product"].unique())
```

```
Total count of Product:
KP281    80
KP481    60
KP781    40
Name: Product, dtype: int64

Unique attributes of Product:
['KP281' 'KP481' 'KP781']
```

Figure 3.7: Value counts and unique attributes of Product column

3.2 Value counts and unique attributes of Age column

Insight: Most number of purchases are made by customers which fall under age range of 23 - 26.

In [13]:
```
print("Total count of Age:")
print(df["Age"].value_counts())

print("\nUnique attributes of Age:")
print(df["Age"].unique())
```

```
Total count of Age:
25    25
23    18
24    12
26    12
28     9
35     8
33     8
30     7
38     7
21     7
22     7
27     7
31     6
34     6
29     6
20     5
40     5
32     4
19     4
48     2
37     2
45     2
47     2
46     1
50     1
18     1
44     1
43     1
41     1
39     1
36     1
42     1
Name: Age, dtype: int64

Unique attributes of Age:
[18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
```

Figure 3.8: Value counts and unique attributes of Age column

3.3 Value counts and unique attributes of Gender column

Insight: Most number of purchases is made by male customers.

In [14]:
```
print("Total count of Gender:")
print(df["Gender"].value_counts())

print("\nUnique attributes of Gender:")
print(df["Gender"].unique())
```

```
Total count of Gender:
Male      104
Female     76
Name: Gender, dtype: int64

Unique attributes of Gender:
['Male' 'Female']
```

Figure 3.9: Value counts and unique attributes of Gender column

- **Similarly checked value counts and unique attributes of Education, MaritalStatus, Usage, Fitness, Income and Miles columns.**

7. **Visual Analysis:** Univariate & Bivariate Analysis

4.1 Checking the effect of Marital Status on the product purchased

Insight: Most number of purchases (all three products) is made by married customers.

In [23]:
```
plt.figure(figsize=(8, 6))

# Countplot for Marital Status and Product Purchased
ax = sns.countplot(x='MaritalStatus', hue='Product', data=df)
plt.title('Effect of Marital Status on Product Purchased')

# Annotate bars
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', xytext=(0, 10), textcoords='offset points')

plt.show()
```

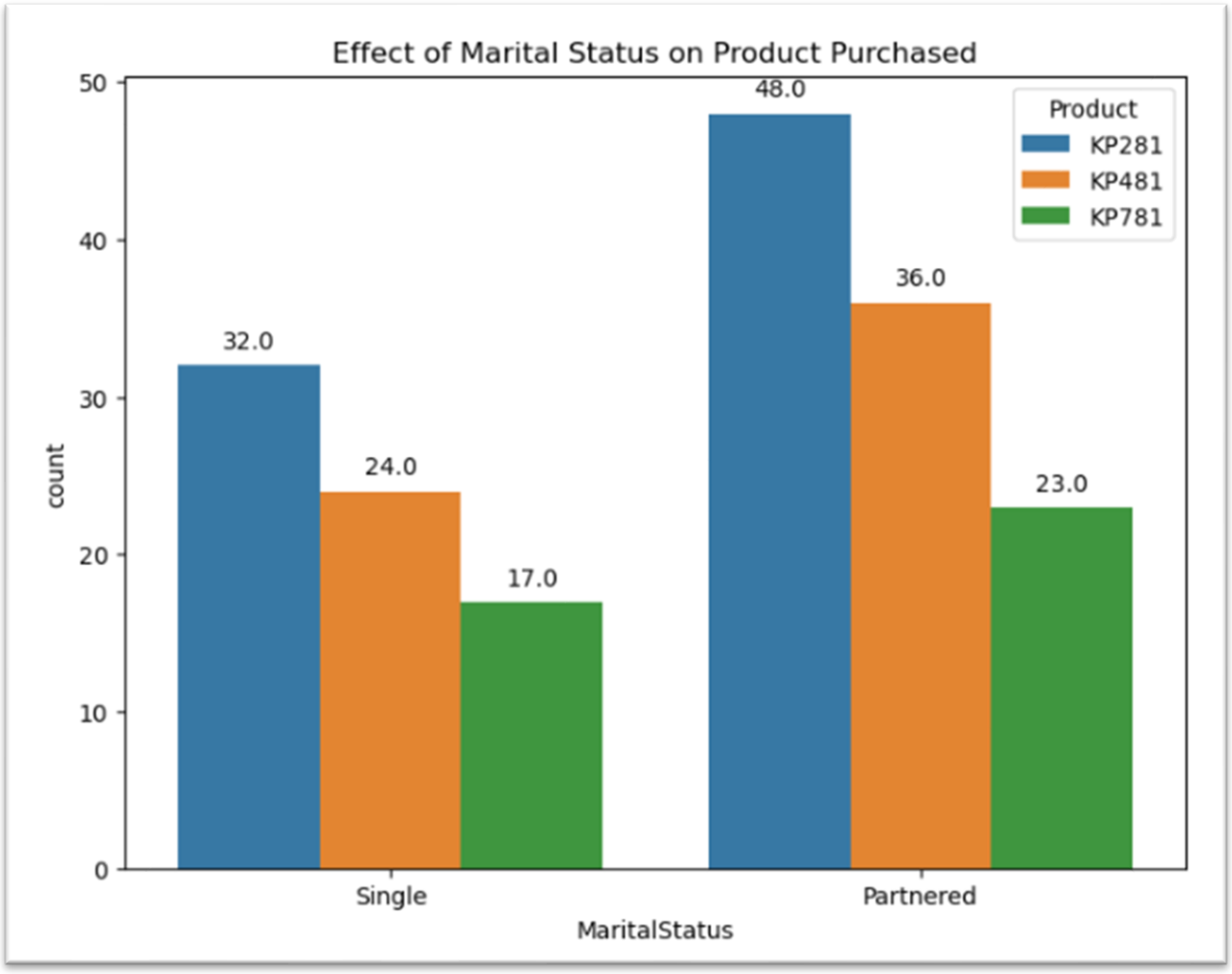


Figure 3.10: Effect of Marital Status on the product purchased

4.2 Checking the effect of Age on the product purchased

Insight: Most number of purchases (all three products) is made by customers aged between 20 - 35 years.

In [24]:
```
plt.figure(figsize=(8, 6))

# Boxplot for Age and Product Purchased
ax = sns.boxplot(x='Product', y='Age', data=df)
plt.title('Effect of Age on Product Purchased')

# Annotate median
for product in df['Product'].unique():
    box_data = df[df['Product'] == product]['Age']
    median = box_data.median()

    vertical_offset = ax.get_ylim()[1] * 0.01  # adjust this based on your preference

    ax.text(df['Product'].unique().tolist().index(product), median + vertical_offset,
            f'Median: {median:.2f}', horizontalalignment='center', color='black', weight='semibold')

plt.show()
```

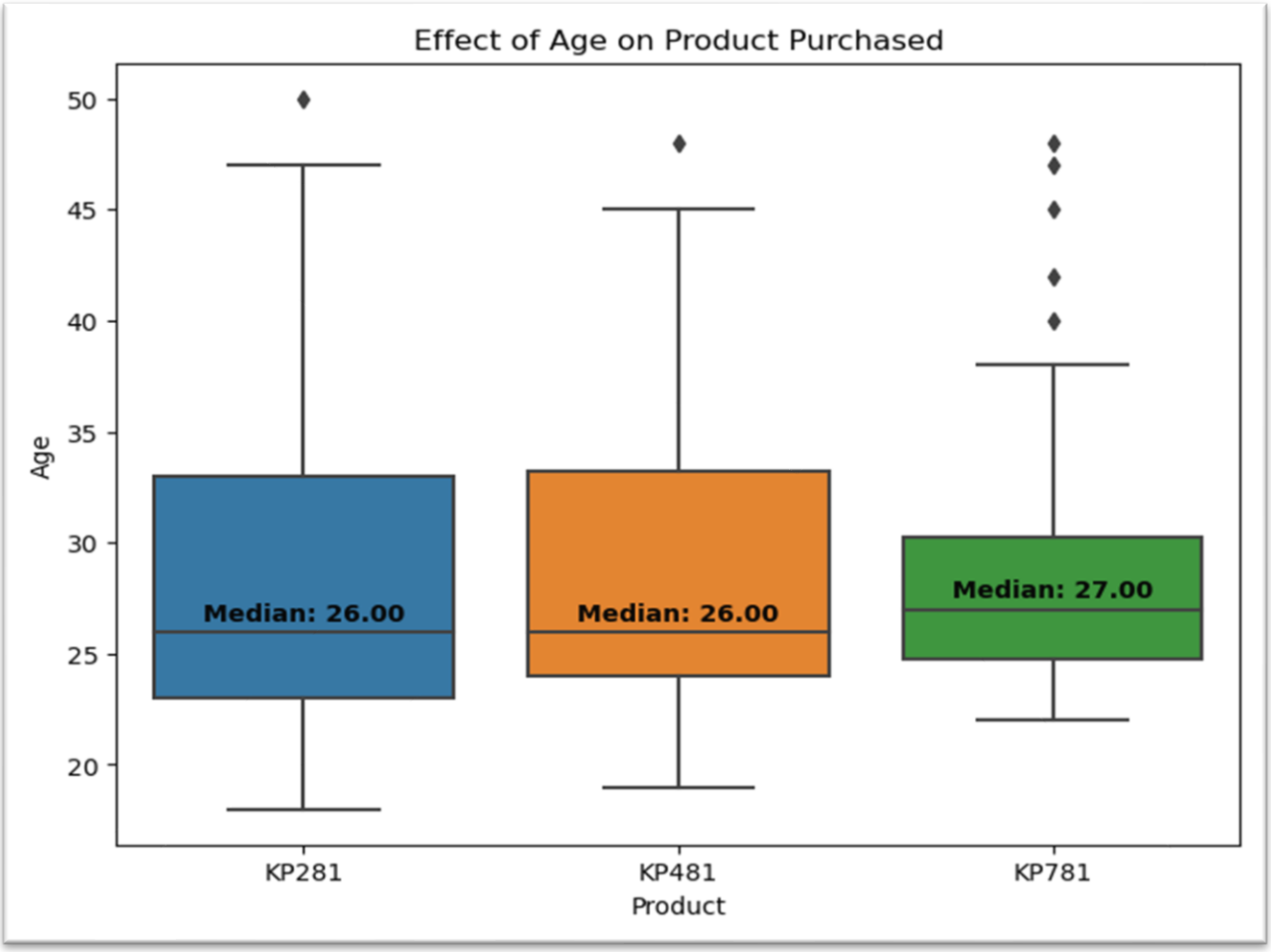


Figure 3.11: Effect of Age on the product purchased

4.3 Checking the effect of Income on the product purchased

Insight: Most of our KP281 and KP481 customers fall in 40,000 - 55,000 dollars annual income bracket. Most of our KP781 customers fall in 60,000 - 90,000 dollars annual income bracket.

In [25]:
```
plt.figure(figsize=(8, 6))

# Boxplot for Income and Product Purchased
ax = sns.boxplot(x='Product', y='Income', data=df)
plt.title('Effect of Income on Product Purchased')

# Annotate median
for product in df['Product'].unique():
    box_data = df[df['Product'] == product]['Income']
    median = box_data.median()

    vertical_offset = ax.get_ylim()[1] * 0.015  # adjust this based on your preference

    ax.text(df['Product'].unique().tolist().index(product), median + vertical_offset,
            f'Median: {median:.2f}', horizontalalignment='center', color='black', weight='semibold')


plt.show()
```

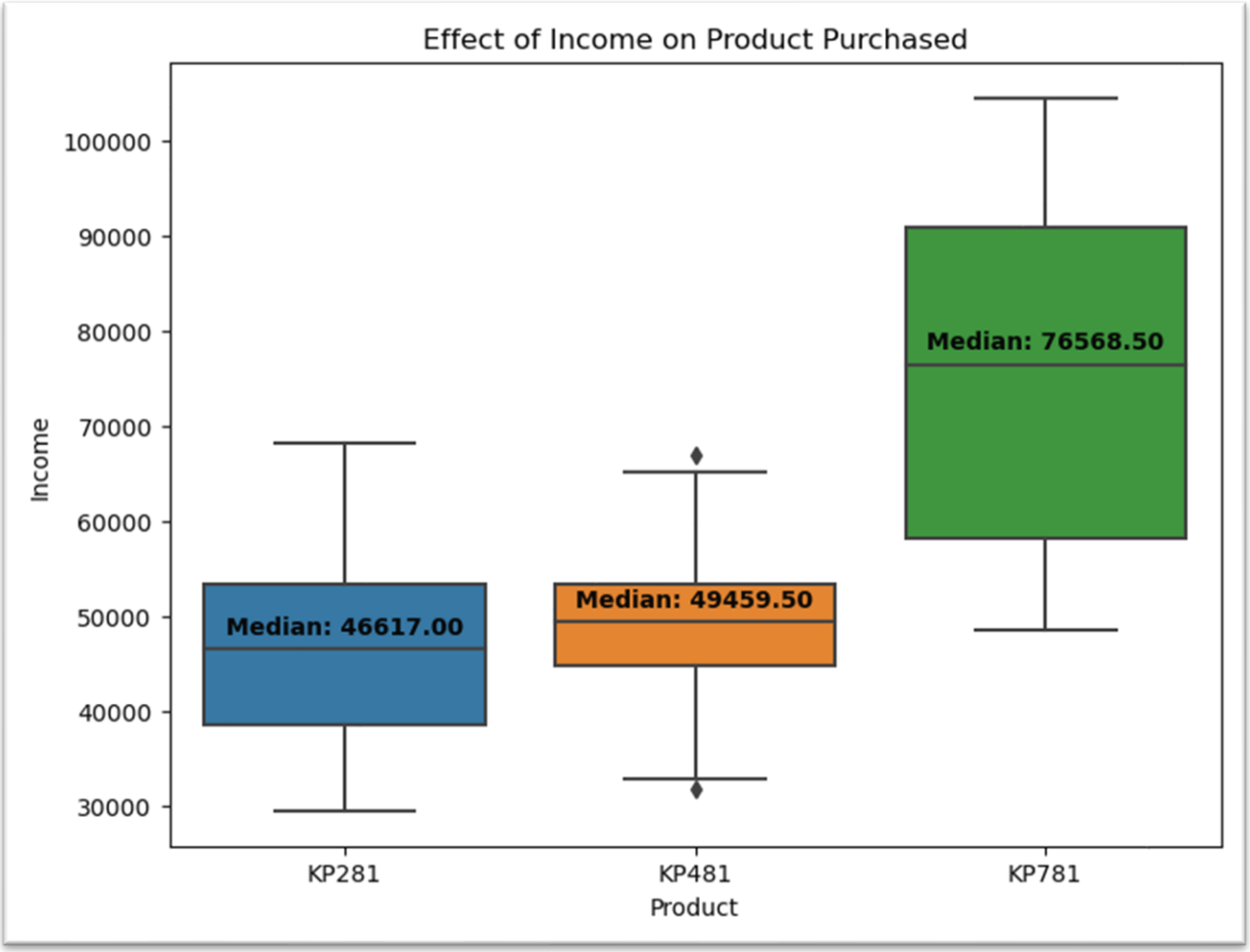


Figure 3.12: Effect of Income on the product purchased

4.4 Checking the effect of Education on the product purchased

Insight: For KP281 and KP481 our customer's education level is moderate. For KP781 our customer's education level is moderate to high.

In [59]:
```
plt.figure(figsize=(8, 6))

# Boxplot for Education and Product Purchased
ax = sns.boxplot(x='Product', y='Education', data=df)
plt.title('Effect of Education on Product Purchased')

# Annotate median
for product in df['Product'].unique():
    box_data = df[df['Product'] == product]['Education']
    median = box_data.median()

    vertical_offset = ax.get_ylim()[1] * 0.02  # adjust this based on your preference

    ax.text(df['Product'].unique().tolist().index(product), median + vertical_offset,
            f'Median: {median:.2f}', horizontalalignment='center', color='black', weight='semibold')

plt.show()
```

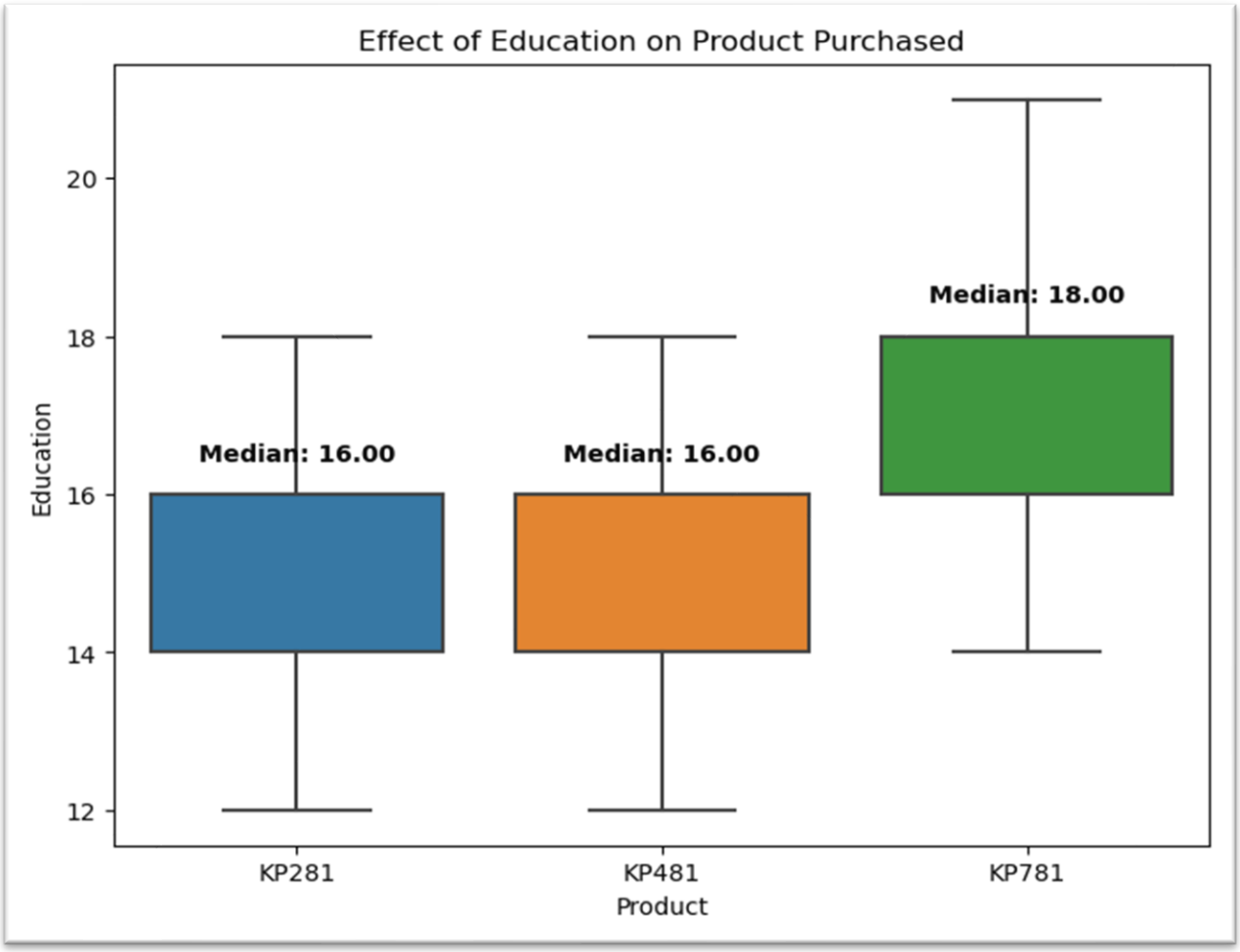


Figure 3.13: Effect of Education on the product purchased

4.5 Univariate Plots - Age Distribution

Insight: Most number of purchases (all three products) is made by customers aged between 20 - 35 years. Average age of our customers is 26 years.

In [67]:

```
plt.figure(figsize=(8, 6))
bins = [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70]
ax = sns.histplot(df['Age'], bins=bins, kde=True)

# Annotate bars with counts
for rect in ax.patches:
    height = rect.get_height()
    ax.annotate(f'{int(height)}', xy=(rect.get_x() + rect.get_width() / 2, height),
                xytext=(0, 5), textcoords='offset points',
                ha='center', va='bottom', fontsize=8)

# Annotate histogram
median_age = df['Age'].median()

# Display median in the top right corner
plt.text(0.95, 0.95, f'Median Age: {median_age:.2f}', transform=ax.transAxes,
         fontsize=12, color='red', ha='right', va='top')

plt.title('Age Distribution - Histogram with Ranges and Median')
plt.show()
```

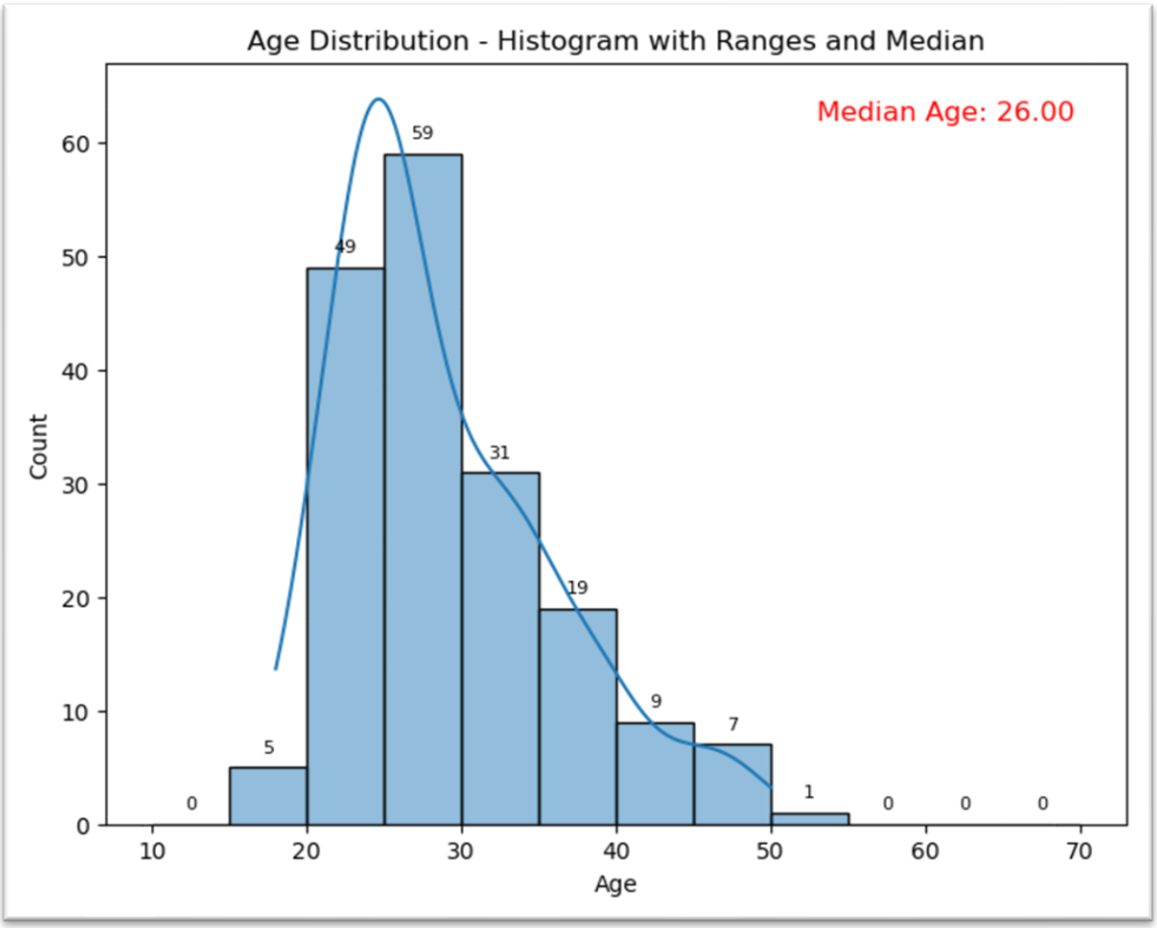


Figure 3.14: Age Distribution

4.6 Univariate Plots - Income Distribution

Insight: Most number of purchases (all three products) is made by customers who fall under 35,000 - 65,000 dollars annual income bracket. Average annual income of our customers is 50,596.5 dollars.

In [73]:

```
plt.figure(figsize=(8, 6))
income_bins = [20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000, 110000, 120000, 130000, 140000]
ax = sns.histplot(df['Income'], bins=income_bins, kde=True, color='orange')

# Annotate bars with counts
for rect in ax.patches:
    height = rect.get_height()
    ax.annotate(f'{int(height)}', xy=(rect.get_x() + rect.get_width() / 2, height),
                xytext=(0, 5), textcoords='offset points',
                ha='center', va='bottom', fontsize=8)

# Annotate histogram
median_income = df['Income'].median()

# Display median in the top right corner
plt.text(0.95, 0.95, f'Median Income: ${median_income:.2f}', transform=ax.transAxes,
         fontsize=12, color='red', ha='right', va='top')

plt.title('Income Distribution - Histogram with Ranges and Median')
plt.show()
```

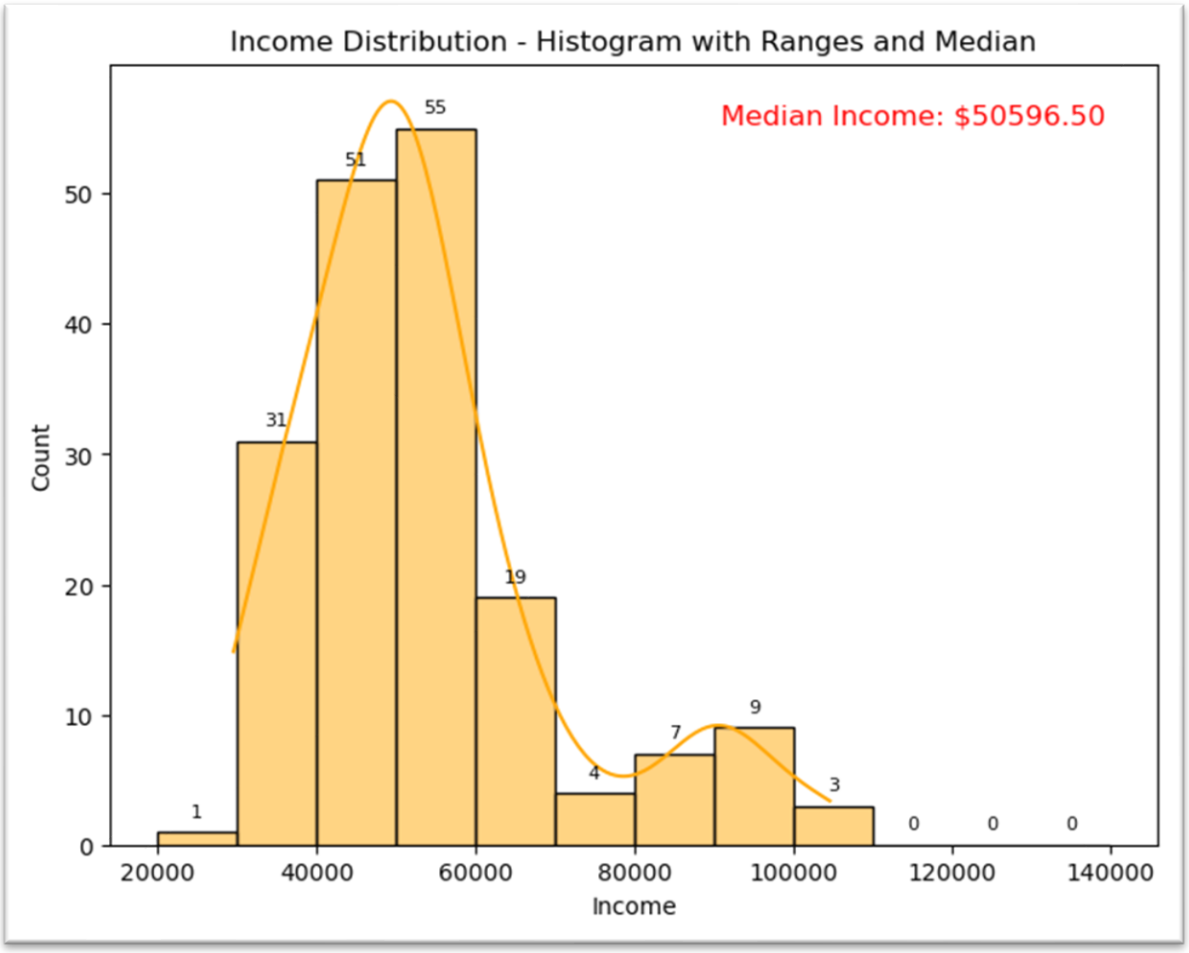


Figure 3.15: Income Distribution

4.7 Univariate Plots - Fitness Distribution

Insight: Most number of purchases (all three products) is made by customers who have moderate level of fitness.

In [29]:
```
plt.figure(figsize=(8, 6))
ax = sns.countplot(x='Fitness', data=df)

# Annotate each bar with counts
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{height}', (p.get_x() + p.get_width() / 2., height),
                ha='center', va='bottom', fontsize=8, color='black')

plt.title('Fitness Distribution - Count Plot')
plt.show()
```

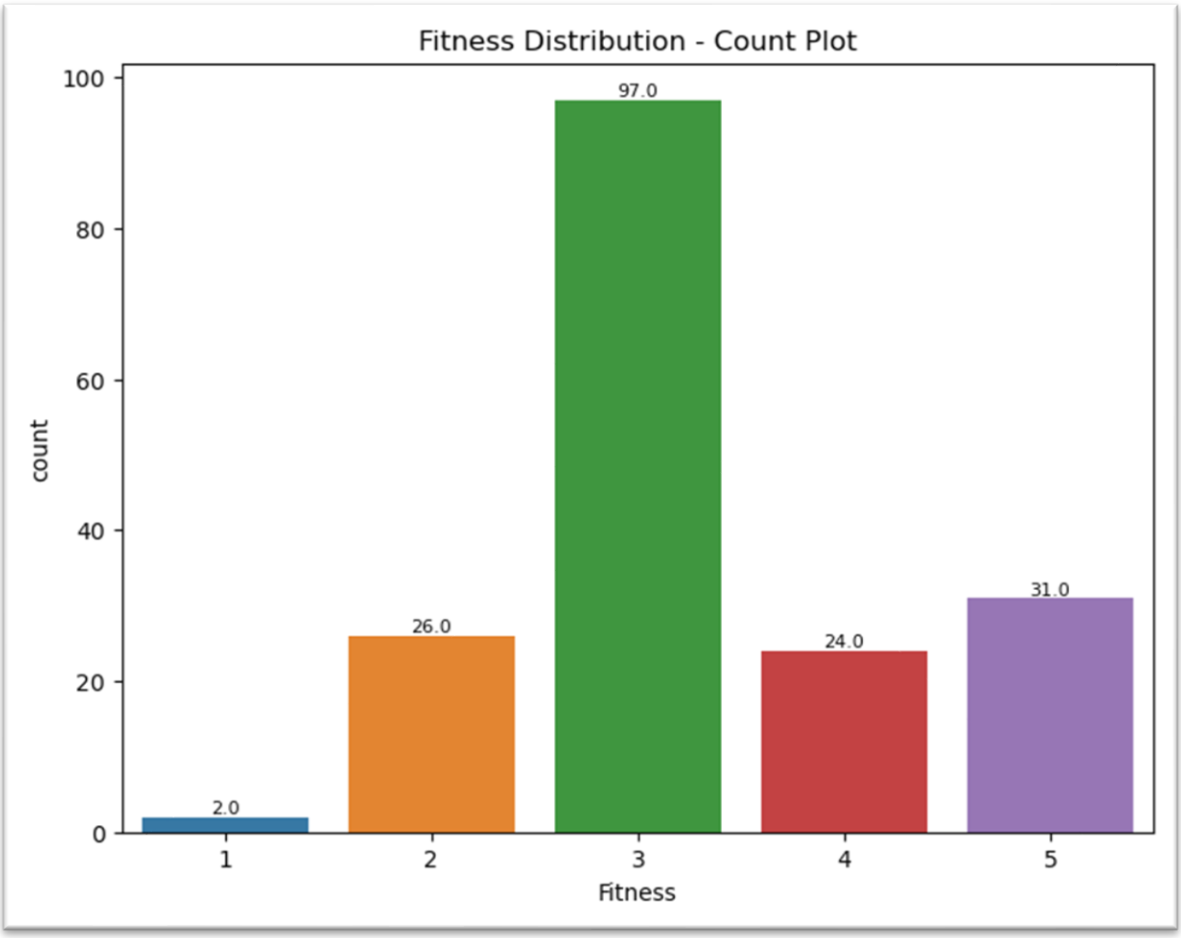


Figure 3.16: Fitness Distribution

4.8 Bivariate Plots - Age vs. Income

In [72]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Income', data=df)

# Median age and income
median_age = df['Age'].median()
median_income = df['Income'].median()

# Display medians in the top right corner
plt.text(0.95, 0.95, f'Median Age: {median_age:.2f}\nMedian Income: ${median_income:.2f}',
         transform=plt.gca().transAxes, fontsize=10, color='red', ha='right', va='top')

plt.title('Age vs. Income - Scatter Plot with Medians')
plt.show()
```

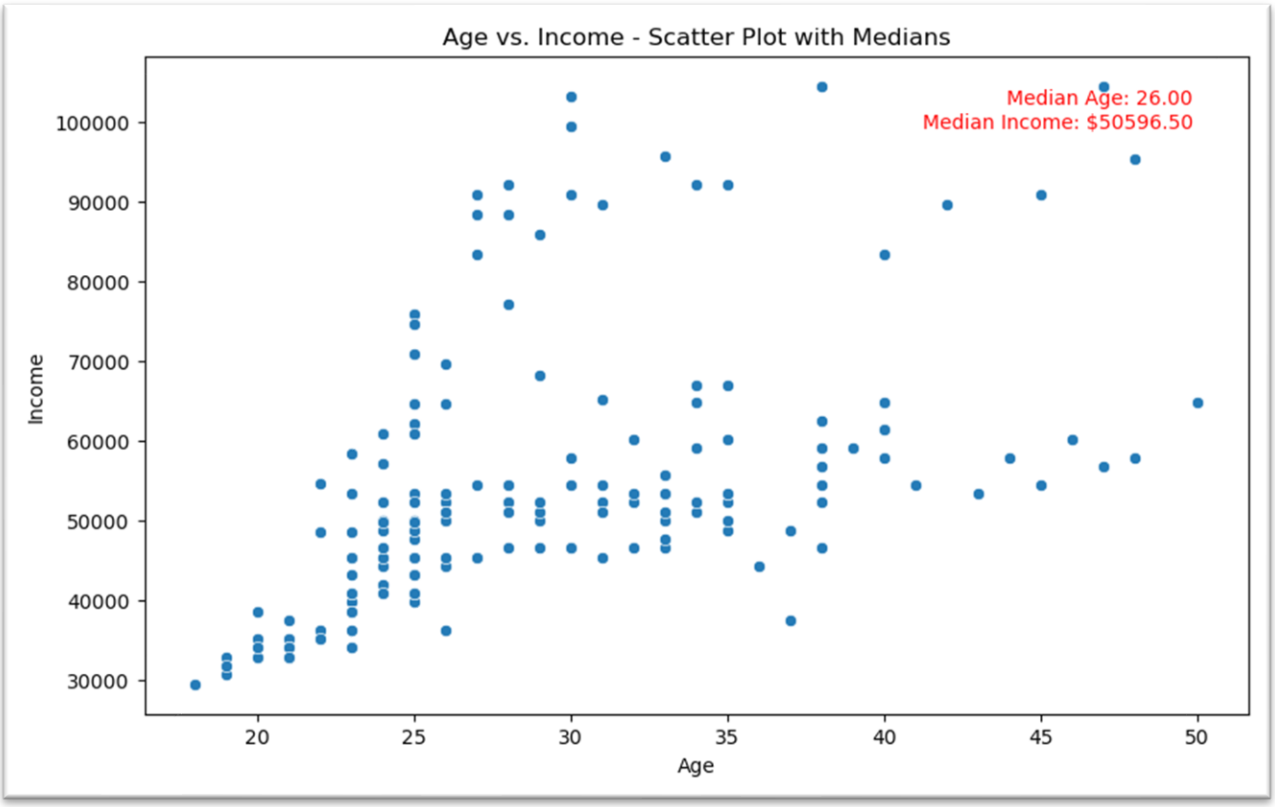


Figure 3.17: Age vs Income- Scatter plot

4.9 Bivariate Plots - Product vs. Usage

Insight: For KP281 our customers are mostly casual users. For KP481 our customers are mostly active to maderate users. For KP781 our customers are mostly athletes or active fitness enthusiasts.

In [31]:
```
plt.figure(figsize=(8, 6))
ax = sns.boxplot(x='Product', y='Usage', data=df)

# Annotate median usage for each box
medians = df.groupby('Product')['Usage'].median()
vertical_offset = df['Usage'].median() * 0.1  # adjust this value for vertical spacing
for xtick, median in zip(ax.get_xticks(), medians):
    ax.text(xtick, median + vertical_offset, f'Median: {median}',
            horizontalalignment='center', color='black', weight='bold')

plt.title('Product vs. Usage - Box Plot')
plt.show()
```

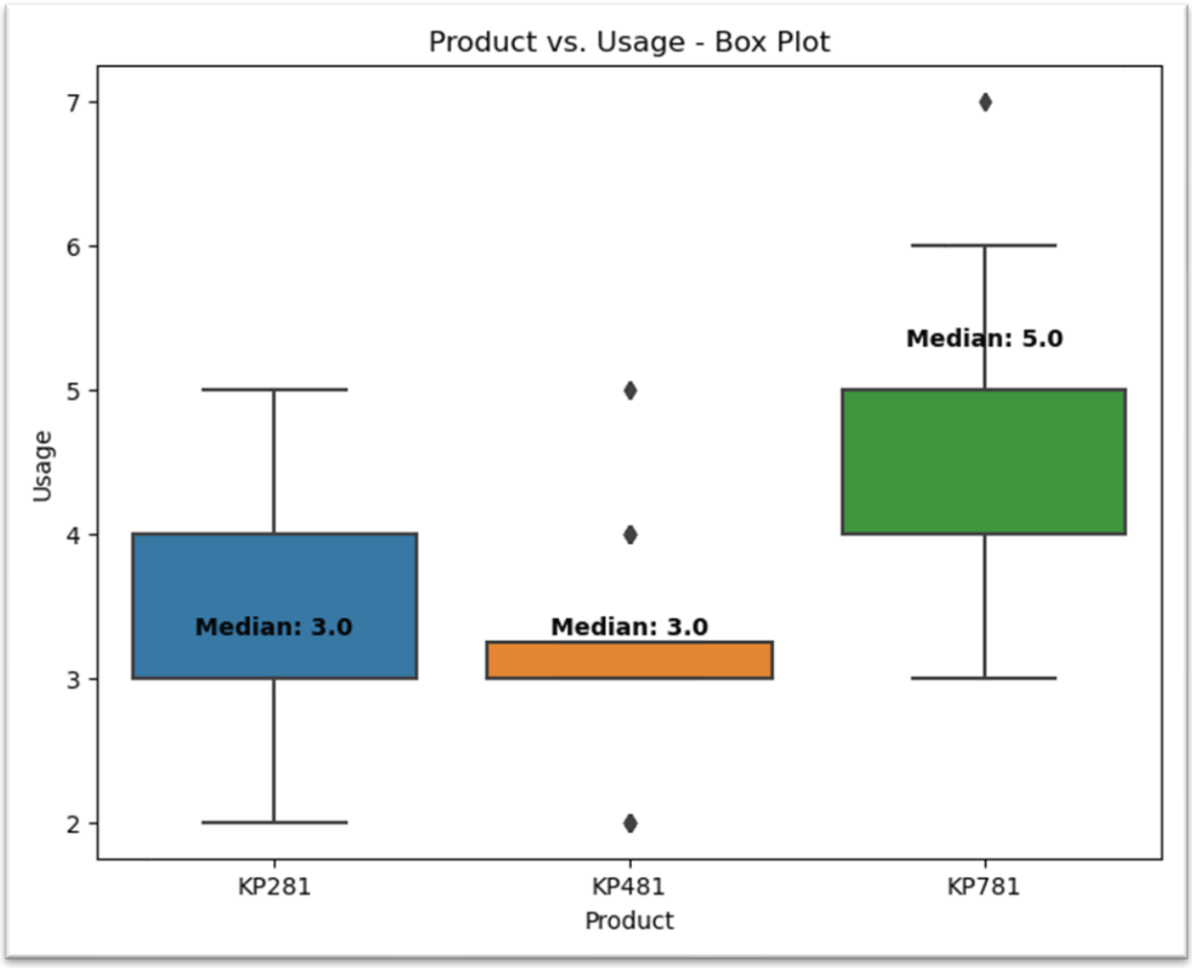


Figure 3.18: Product vs Usage- Box plot

4.10 Bivariate Plots - Gender vs. Fitness

Insight: Most of our customers have moderate level of fitness regardless of gender.

In [32]:
```
plt.figure(figsize=(8, 6))
ax = sns.boxplot(x='Gender', y='Fitness', data=df)

# Annotate median usage for each box
medians = df.groupby('Product')['Usage'].median()
vertical_offset = df['Usage'].median() * 0.1  # adjust this value for vertical spacing
for xtick, median in zip(ax.get_xticks(), medians):
    ax.text(xtick, median + vertical_offset, f'Median: {median}',
            horizontalalignment='center', color='black', weight='bold')

plt.title('Gender vs. Fitness - Box Plot')
plt.show()
```

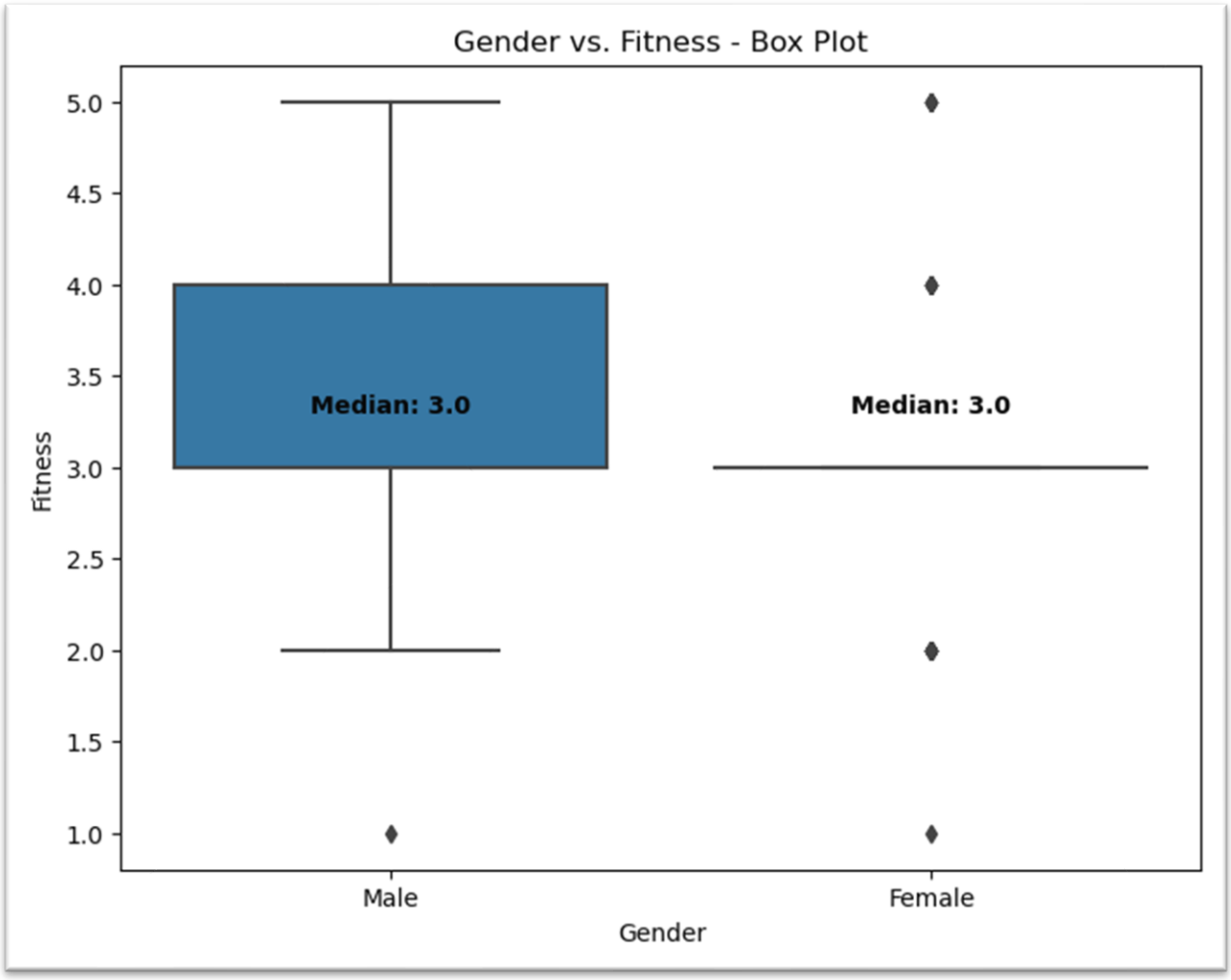


Figure 3.19: Gender vs Fitness- Box plot

4.11 Checking Correlation among Different Factors

In [33]:
```
# Calculate correlation matrix for numeric columns
numeric_columns = df.select_dtypes(include='number').columns
correlation_matrix = df[numeric_columns].corr()

# Plotting the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

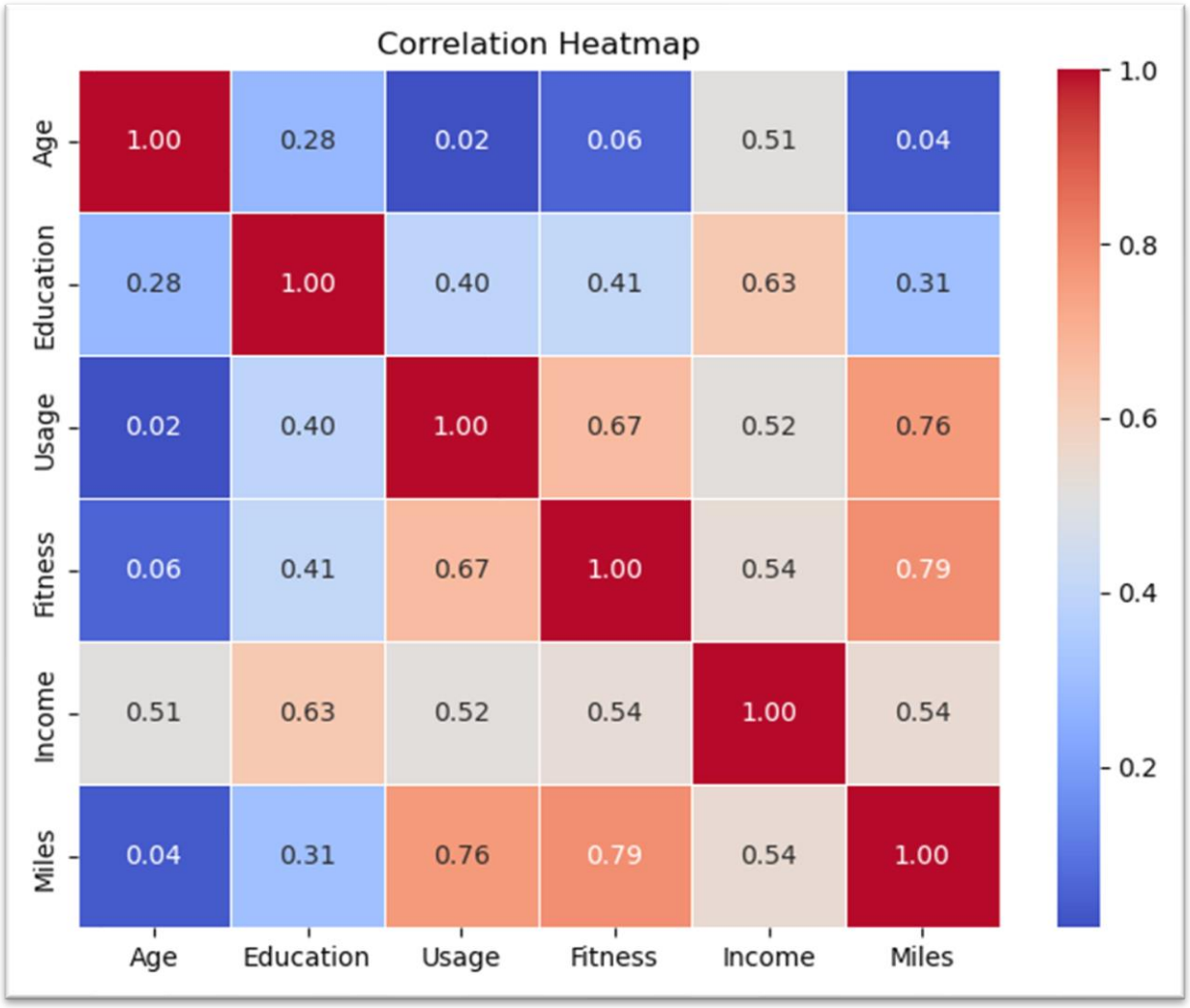


Figure 3.20: Correlation Heatmap

8. **Marginal and Conditional Probability:**

    1. Marginal probability is the probability of a single event without considering any other events. In this case, we have calculated the:

        1. Marginal Probability of each product being purchased - P(KP281), P(K4281), P(KP781)
        2. Marginal Probability of marital status - P(married), P(single)
        3. Marginal Probability of education
        4. Marginal Probability of fitness

In [34]:
```
# Marginal Probability of each product being purchased
marginal_prob_product = df['Product'].value_counts(normalize=True)
print("\nMarginal Probability of each product being purchased:")
print(marginal_prob_product)

# Marginal Probability of MaritalStatus
marginal_prob_marital = df['MaritalStatus'].value_counts(normalize=True)
print("\n\nMarginal Probability of MaritalStatus:")
print(marginal_prob_marital)

# Marginal Probability of education
marginal_prob_education = df['Education'].value_counts(normalize=True)
print("\n\nMarginal Probability of education:")
print(marginal_prob_education)

# Marginal Probability of fitness
marginal_prob_fitness = df['Fitness'].value_counts(normalize=True)
print("\n\nMarginal Probability of fitness:")
print(marginal_prob_fitness)
```

```
Marginal Probability of each product being purchased:
KP281    0.444444
KP481    0.333333
KP781    0.222222
Name: Product, dtype: float64


Marginal Probability of MaritalStatus:
Partnered    0.594444
Single       0.405556
Name: MaritalStatus, dtype: float64


Marginal Probability of education:
16    0.472222
14    0.305556
18    0.127778
15    0.027778
13    0.027778
12    0.016667
21    0.016667
20    0.005556
Name: Education, dtype: float64


Marginal Probability of fitness:
3    0.538889
5    0.172222
2    0.144444
4    0.133333
1    0.011111
Name: Fitness, dtype: float64
```

Figure 3.21: Marginal Probability

9. **Marginal and Conditional Probability:**

Conditional probability is the probability of an event occurring given that another event has already occurred. In this case, we have calculated the:

1. Conditional Probability of purchasing each product given the gender - P(KP281 | male), P(KP281 | female), P(KP481 | male), P(KP481 | female), P(KP781 | male), P(KP781 | female)
2. Conditional Probability of purchasing each product given marital status - P(KP281 | married), P(KP281 | single), P(KP481 | married), P(KP481 | single), P(KP781 | married), P(KP281 | single)
3. Conditional Probability of purchasing each product given education
4. Conditional Probability of purchasing each product given fitness

```
In [35]: # Conditional Probability of purchasing each product given gender
         conditional_prob_gender_product = pd.crosstab(index=df['Gender'], columns=df['Product'], normalize='index')
         print("\nConditional Probability of purchasing each product given gender:")
         print(conditional_prob_gender_product)

         # Conditional Probability of purchasing each product given marital status
         conditional_prob_marital_product = pd.crosstab(index=df['MaritalStatus'], columns=df['Product'], normalize='index')
         print("\n\nConditional Probability of purchasing each product given marital status:")
         print(conditional_prob_marital_product)

         # Conditional Probability of purchasing each product given education
         conditional_prob_education_product = pd.crosstab(index=df['Education'], columns=df['Product'], normalize='index')
         print("\n\nConditional Probability of purchasing each product given education:")
         print(conditional_prob_education_product)

         # Conditional Probability of purchasing each product given fitness
         conditional_prob_fitness_product = pd.crosstab(index=df['Fitness'], columns=df['Product'], normalize='index')
         print("\n\nConditional Probability of purchasing each product given fitness status:")
         print(conditional_prob_fitness_product)
```

```
Conditional Probability of purchasing each product given gender:
Product      KP281     KP481     KP781
Gender
Female    0.526316  0.381579  0.092105
Male      0.384615  0.298077  0.317308


Conditional Probability of purchasing each product given marital status:
Product           KP281     KP481     KP781
MaritalStatus
Partnered      0.448598  0.336449  0.214953
Single         0.438356  0.328767  0.232877


Conditional Probability of purchasing each product given education:
Product        KP281     KP481     KP781
Education
12          0.666667  0.333333  0.000000
13          0.600000  0.400000  0.000000
14          0.545455  0.418182  0.036364
15          0.800000  0.200000  0.000000
16          0.458824  0.364706  0.176471
18          0.086957  0.086957  0.826087
20          0.000000  0.000000  1.000000
21          0.000000  0.000000  1.000000


Conditional Probability of purchasing each product given fitness status:
Product      KP281     KP481     KP781
Fitness
1         0.500000  0.500000  0.000000
2         0.538462  0.461538  0.000000
3         0.556701  0.402062  0.041237
4         0.375000  0.333333  0.291667
5         0.064516  0.000000  0.935484
```

Figure 3.22: Conditional Probability

10. **For each AeroFit treadmill product, we have constructed two-way contingency tables and computed all conditional and marginal probabilities**

```
In [36]: # Contingency table for Product KP281
         contingency_table_KP281 = pd.crosstab(index = df['Gender'],
                                               columns = df['MaritalStatus'],
                                               margins = True,
                                               margins_name = 'Total')
         conditional_prob_KP281 = contingency_table_KP281 / contingency_table_KP281.loc['Total', 'Total']

         # Contingency table for Product KP481
         contingency_table_KP481 = pd.crosstab(index = df['Gender'],
                                               columns = df['MaritalStatus'],
                                               margins = True,
                                               margins_name = 'Total')
         conditional_prob_KP481 = contingency_table_KP481 / contingency_table_KP481.loc['Total', 'Total']

         # Contingency table for Product KP781
         contingency_table_KP781 = pd.crosstab(index = df['Gender'],
                                               columns = df['MaritalStatus'],
                                               margins = True,
                                               margins_name = 'Total')
         conditional_prob_KP781 = contingency_table_KP781 / contingency_table_KP781.loc['Total', 'Total']

         # Display results for KP281
         print("Conditional Probabilities for KP281:")
         print(conditional_prob_KP281)
         print("\nMarginal Probabilities for KP281:")
         print(contingency_table_KP281 / len(df))  # Marginal probabilities

         # Display results for KP481
         print("\nConditional Probabilities for KP481:")
         print(conditional_prob_KP481)
         print("\nMarginal Probabilities for KP481:")
         print(contingency_table_KP481 / len(df))  # Marginal probabilities

         # Display results for KP781
         print("\nConditional Probabilities for KP781:")
         print(conditional_prob_KP781)
         print("\nMarginal Probabilities for KP781:")
         print(contingency_table_KP781 / len(df))  # Marginal probabilities
```

```
Conditional Probabilities for KP281:
MaritalStatus  Partnered    Single     Total
Gender
Female          0.255556  0.166667  0.422222
Male            0.338889  0.238889  0.577778
Total           0.594444  0.405556  1.000000

Marginal Probabilities for KP281:
MaritalStatus  Partnered    Single     Total
Gender
Female          0.255556  0.166667  0.422222
Male            0.338889  0.238889  0.577778
Total           0.594444  0.405556  1.000000

Conditional Probabilities for KP481:
MaritalStatus  Partnered    Single     Total
Gender
Female          0.255556  0.166667  0.422222
Male            0.338889  0.238889  0.577778
Total           0.594444  0.405556  1.000000

Marginal Probabilities for KP481:
MaritalStatus  Partnered    Single     Total
Gender
Female          0.255556  0.166667  0.422222
Male            0.338889  0.238889  0.577778
Total           0.594444  0.405556  1.000000

Conditional Probabilities for KP781:
MaritalStatus  Partnered    Single     Total
Gender
Female          0.255556  0.166667  0.422222
Male            0.338889  0.238889  0.577778
Total           0.594444  0.405556  1.000000

Marginal Probabilities for KP781:
MaritalStatus  Partnered    Single     Total
Gender
Female          0.255556  0.166667  0.422222
Male            0.338889  0.238889  0.577778
Total           0.594444  0.405556  1.000000
```

Figure 3.23: Two-way contingency tables for all conditional and marginal probabilities

5.4 What is P(Single | KP281) i.e. product bought is KP281 what is the probability that this person is single?

```
In [70]: df[df.Product == "KP281"]["MaritalStatus"].value_counts(normalize = True)

Out[70]: Partnered    0.6
         Single       0.4
         Name: MaritalStatus, dtype: float64
```

5.5 What is P(Single | KP481) i.e. product bought is KP481 what is the probability that this person is single?

```
In [71]: df[df.Product == "KP481"]["MaritalStatus"].value_counts(normalize = True)

Out[71]: Partnered    0.6
         Single       0.4
         Name: MaritalStatus, dtype: float64
```

5.6 What is P(Single | KP781) i.e. product bought is KP781 what is the probability that this person is single?

```
In [72]: df[df.Product == "KP781"]["MaritalStatus"].value_counts(normalize = True)

Out[72]: Partnered    0.575
         Single       0.425
         Name: MaritalStatus, dtype: float64
```

5.7 What is P(Purchase | Male) i.e. for each product bought what is the probability that a male buys it?

```
In [79]: df[df.Gender=="Male"]["Product"].value_counts(normalize = True)

Out[79]: KP281    0.384615
         KP781    0.317308
         KP481    0.298077
         Name: Product, dtype: float64
```

5.8 What is P(Purchase | Female) i.e. for each product bought what is the probability that a female buys it?

```
In [80]: df[df.Gender == "Female"]["Product"].value_counts(normalize = True)

Out[80]: KP281    0.526316
         KP481    0.381579
         KP781    0.092105
         Name: Product, dtype: float64
```

Figure 3.24: Some other important conditional probabilities (1)

5.9 Probability of a Male and Female Customers Buying a KP281 Treadmill

Insight: P(male buying KP281) = 0.38. P(female buying KP281) = 0.53. We note that there are very high chances that a female will buy KP281 as compared to KP481 and KP781.

In [76]:
```
male_kp281_prob = df[(df['Gender'] == 'Male') & (df['Product'] == 'KP281')].shape[0] / df[df['Gender'] == 'Male'].shape[0]
female_kp281_prob = df[(df['Gender'] == 'Female')&(df['Product'] == 'KP281')].shape[0]/df[df['Gender'] == 'Female'].shape[0]

print(f"\nProbability of a Male customer buying a KP281 treadmill: {male_kp281_prob:.2f}")
print(f"Probability of a Female customer buying a KP281 treadmill: {female_kp281_prob:.2f}")
```

```
Probability of a Male customer buying a KP281 treadmill: 0.38
Probability of a Female customer buying a KP281 treadmill: 0.53
```

5.10 Probability of a Male and Female Customer Buying a KP481 Treadmill

Insight: P(male buying KP481) = 0.30. P(female buying KP481) = 0.38.

In [75]:
```
male_kp481_prob = df[(df['Gender'] == 'Male') & (df['Product'] == 'KP481')].shape[0] / df[df['Gender'] == 'Male'].shape[0]
female_kp481_prob = df[(df['Gender'] == 'Female')&(df['Product'] == 'KP481')].shape[0]/df[df['Gender'] == 'Female'].shape[0]

print(f"\nProbability of a Male customer buying a KP481 treadmill: {male_kp481_prob:.2f}")
print(f"Probability of a Female customer buying a KP481 treadmill: {female_kp481_prob:.2f}")
```

```
Probability of a Male customer buying a KP481 treadmill: 0.30
Probability of a Female customer buying a KP481 treadmill: 0.38
```

Figure 3.25: Some other important conditional probabilities (2)

5.11 Probability of a Male and Female Customer Buying a KP781 Treadmill

Insight: P(male buying KP781) = 0.32. P(female buying KP781) = 0.09. We note that there are very less chances that a female will buy KP781 as compared to KP281 and KP481.

In [74]:
```
male_kp781_prob = df[(df['Gender'] == 'Male') & (df['Product'] == 'KP781')].shape[0] / df[df['Gender'] == 'Male'].shape[0]
female_kp781_prob = df[(df['Gender'] == 'Female')&(df['Product'] == 'KP781')].shape[0]/df[df['Gender'] == 'Female'].shape[0]

print(f"\nProbability of a Male customer buying a KP781 treadmill: {male_kp781_prob:.2f}")
print(f"Probability of a Female customer buying a KP781 treadmill: {female_kp781_prob:.2f}")
```

```
Probability of a Male customer buying a KP781 treadmill: 0.32
Probability of a Female customer buying a KP781 treadmill: 0.09
```

5.12 Overall probability of purchase for each treadmill

Insight: P(KP281) = 0.44, P(KP481) = 0.33, P(K781) = 0.22.

In [73]:
```
total_customers = len(df)

# Probability of Purchase for KP281
prob_purchase_kp281 = len(df[df['Product'] == 'KP281']) / total_customers

# Probability of Purchase for KP481
prob_purchase_kp481 = len(df[df['Product'] == 'KP481']) / total_customers

# Probability of Purchase for KP781
prob_purchase_kp781 = len(df[df['Product'] == 'KP781']) / total_customers

print(f"Overall Probability of Purchase for KP281: {prob_purchase_kp281:.2f}")
print(f"Overall Probability of Purchase for KP481: {prob_purchase_kp481:.2f}")
print(f"Overall Probability of Purchase for KP781: {prob_purchase_kp781:.2f}")
```

```
Overall Probability of Purchase for KP281: 0.44
Overall Probability of Purchase for KP481: 0.33
Overall Probability of Purchase for KP781: 0.22
```

Figure 3.26: Some other important conditional probabilities (3)

11. **Outlier Detection:**

6.1 Detecting Outliers using describe method

In [41]:
```
# Use describe method to get summary statistics
summary_stats = df.describe()

# Calculate the interquartile range (IQR)
Q1 = summary_stats.loc['25%']
Q3 = summary_stats.loc['75%']
IQR = Q3 - Q1

# Define lower and upper bounds to identify outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Align the DataFrame and perform the comparison
df_aligned, lower_bound_aligned = df.align(lower_bound, axis=1, join='outer')
df_aligned, upper_bound_aligned = df.align(upper_bound, axis=1, join='outer')

# Identify outliers
outliers = (df_aligned < lower_bound_aligned) | (df_aligned > upper_bound_aligned)

# Display the outliers
print("Outliers:")
print(outliers.sum())
```

```
Outliers:
Age              5
Education        4
Fitness          2
Gender           0
Income          19
MaritalStatus    0
Miles           13
Product          0
Usage            9
dtype: int64
```

Figure 3.27: Detecting Outliers using describe method

In [42]:
```
# Set up the matplotlib figure
plt.figure(figsize=(6, 4))

# Create a box plot for the 'Age' column
sns.boxplot(x=df['Age'])
plt.title('Detecting outliers using Box Plot for Age attribute')
plt.show()
```

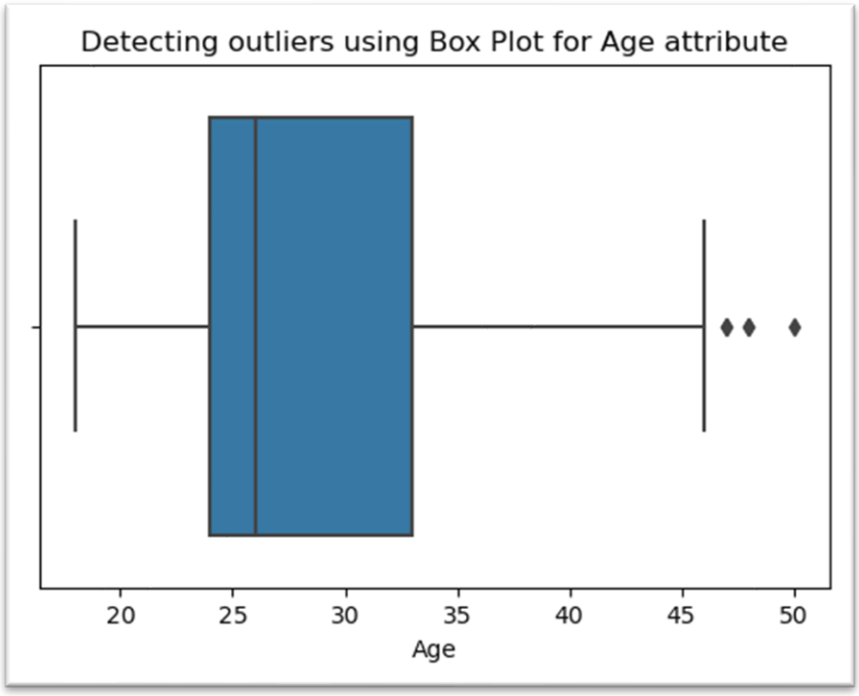
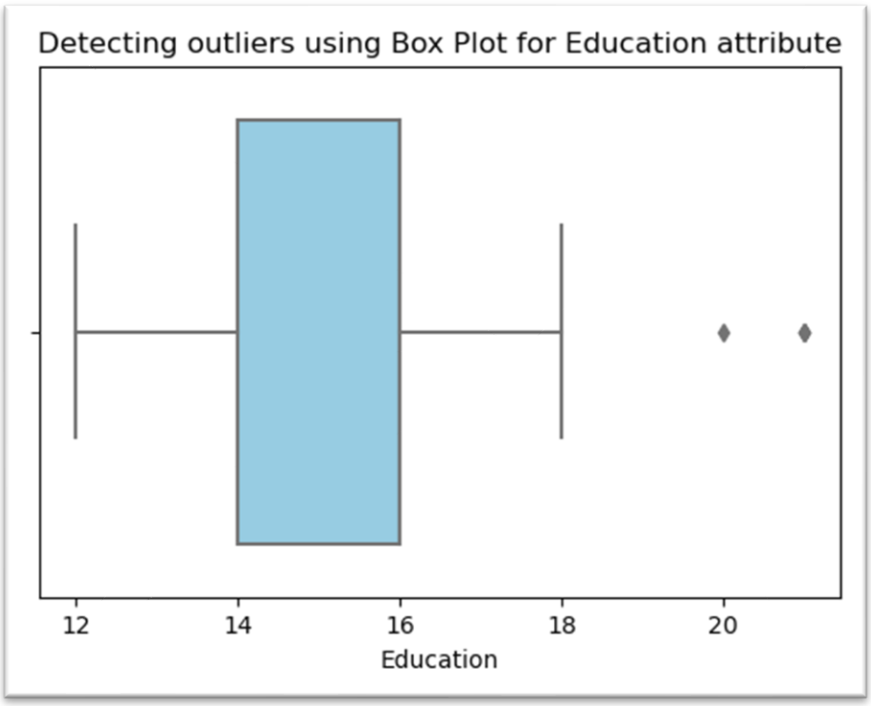
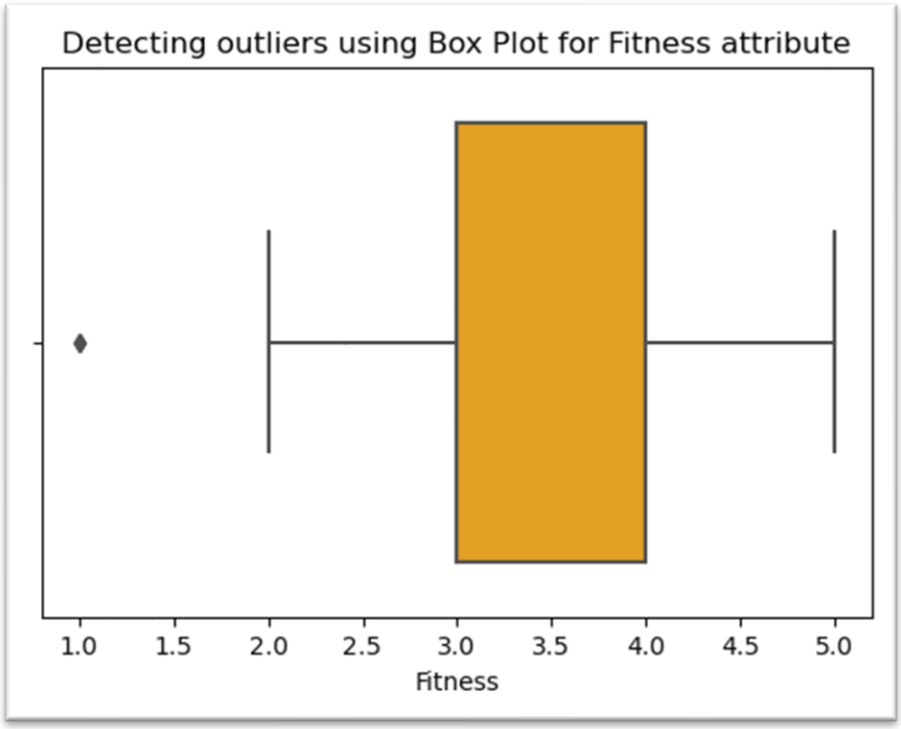
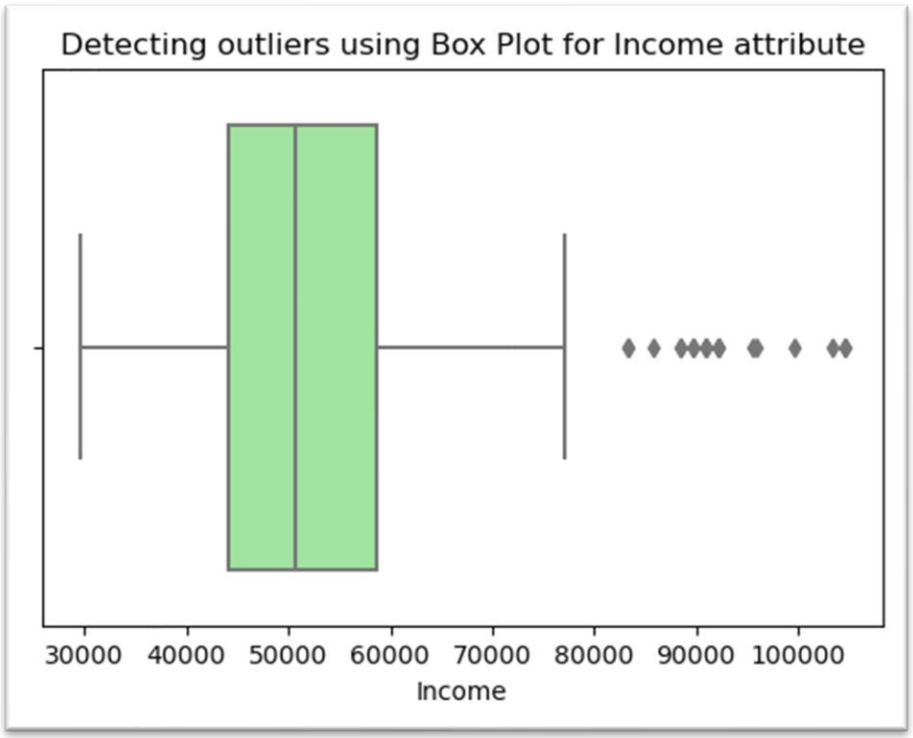


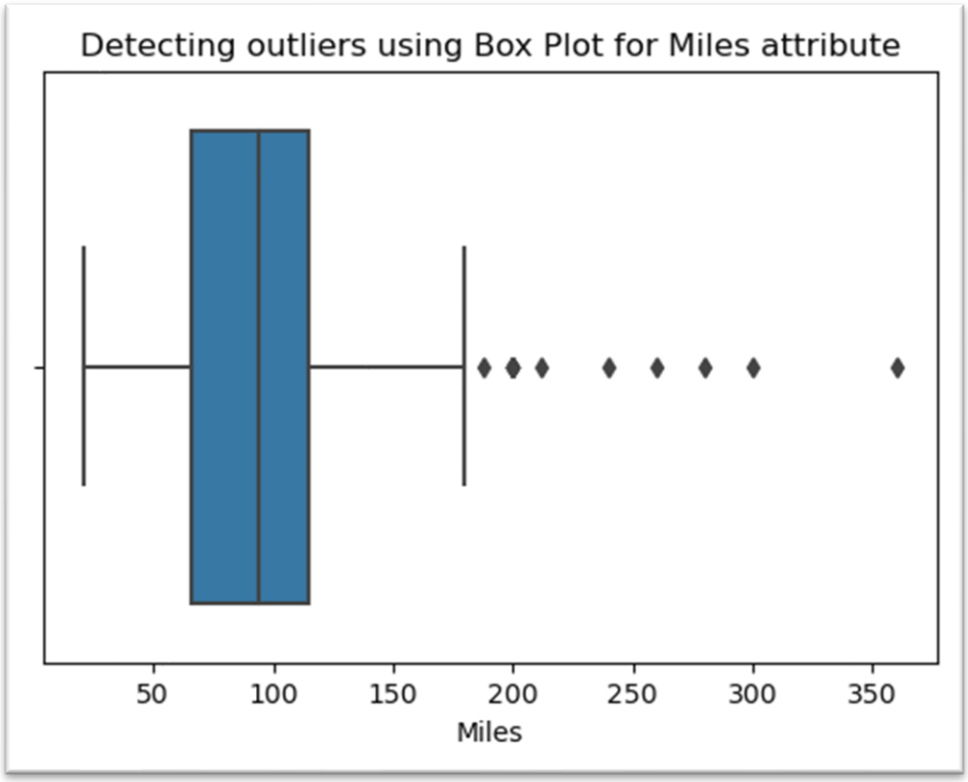
Figure 3.28: Detecting Outliers using Box Plot for Age attribute

- **Similarly Detecting Outliers using Box Plot for all other attributes:**




Detecting outliers using Box Plot for Fitness attribute
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Fitness


Detecting outliers using Box Plot for Income attribute
30000
40000
50000
60000
70000
80000
90000
100000
Income



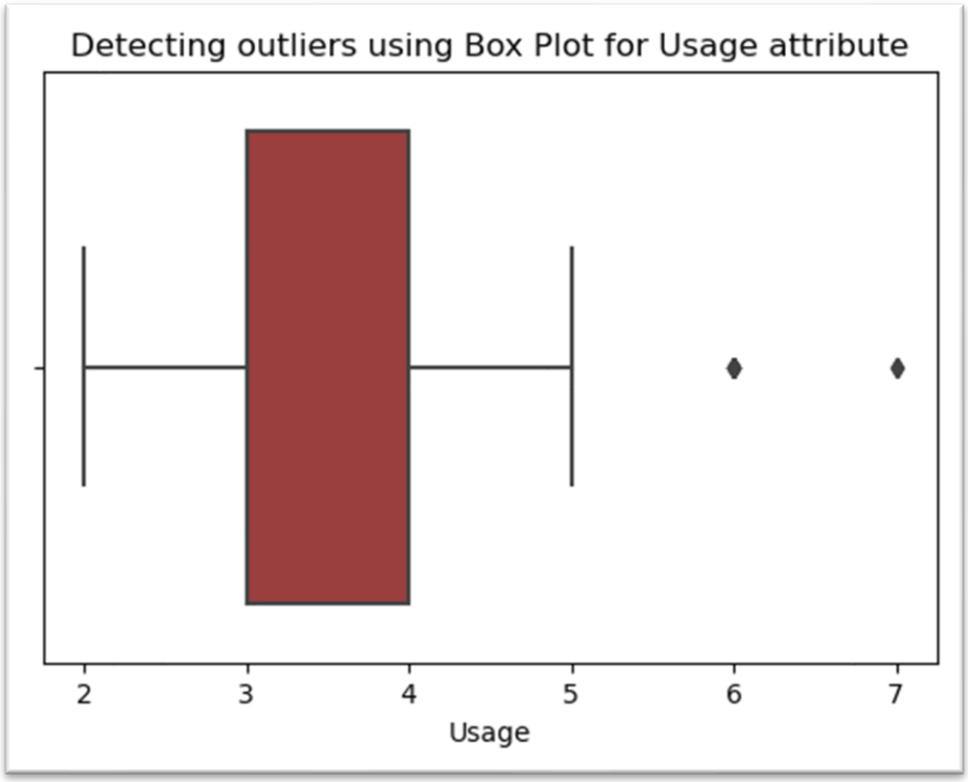


Figure 3.29: Detecting Outliers using Box Plot for all other attributes

12. Customer Profiling - Categorization of users:

1. Customer profiling involves categorizing users based on certain characteristics to better understand and target specific customer segments. We have considered creating segments based on usage patterns and fitness levels.
2. INSIGHTS: Most of our customers are active, moderate fitness users, followed by casual users and athletes.

```
In [21]: # Create segments based on Usage and Fitness
         # We can customize these criteria based on your business goals
         df['Segment'] = 'Undecided'
         df.loc[(df['Usage'] >= 3) & (df['Fitness'] >= 4), 'Segment'] = 'Active Fitness Enthusiasts'
         df.loc[(df['Usage'] >= 3) & (df['Fitness'] < 4), 'Segment'] = 'Active, Moderate Fitness'
         df.loc[df['Usage'] < 3, 'Segment'] = 'Casual Users'

         # Visualize the segments
         plt.figure()
         sns.scatterplot(x='Usage', y='Fitness', hue='Segment', data=df, palette='bright', s=200)
         plt.title('Customer Segmentation based on Usage and Fitness')
         plt.show()

         # Analyze the segments
         segment_analysis = df.groupby('Segment').agg({
             'Usage': ['mean', 'std'],
             'Fitness': ['mean', 'std'],
             'Product': 'count'
         }).reset_index()

         segment_analysis.columns = ['Segment', 'Avg_Usage', 'Std_Usage', 'Avg_Fitness', 'Std_Fitness', 'Customer_Count']

         print("\nSegment Analysis:")
         print(segment_analysis)
```

```
Segment Analysis:
                    Segment  Avg_Usage  Std_Usage  Avg_Fitness  Std_Fitness  \
0  Active Fitness Enthusiasts   4.436364   1.067424     4.563636     0.500505
1    Active, Moderate Fitness   3.391304   0.533628     2.847826     0.390401
2                Casual Users   2.000000   0.000000     2.515152     0.565752

   Customer_Count
0              55
1              92
2              33
```

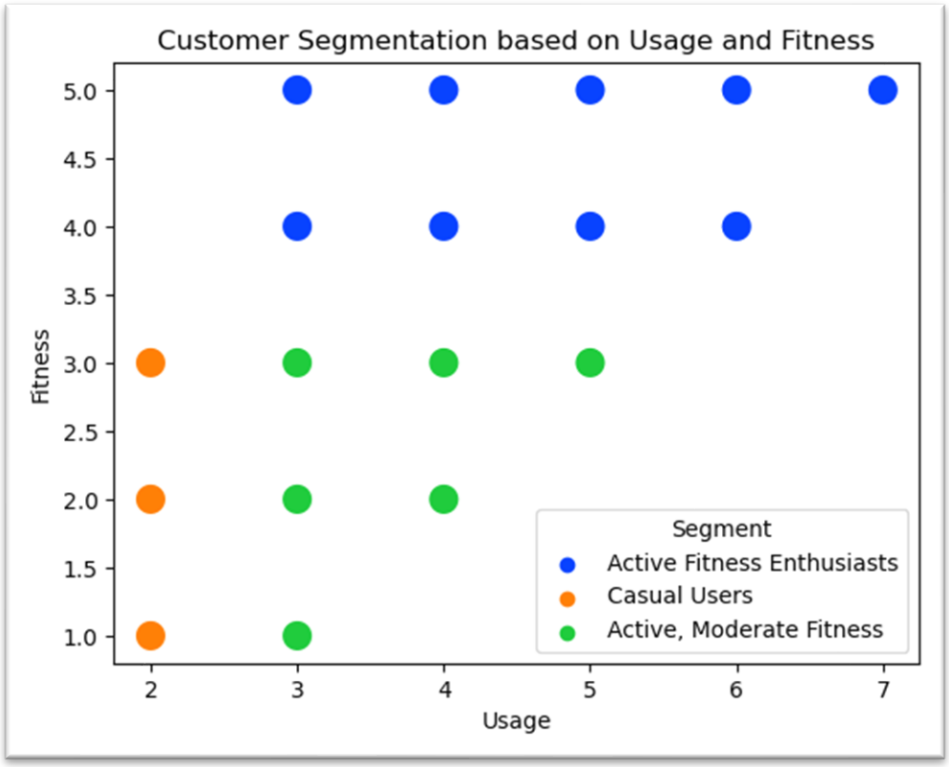


Figure 3.30: Customer Profiling - Categorization of users

7.2 Customer Profiles for Each Product

In [22]:
```
# Filter the DataFrame for each product
kp281_data = df[df['Product'] == 'KP281']
kp481_data = df[df['Product'] == 'KP481']
kp781_data = df[df['Product'] == 'KP781']

# Define a function to calculate customer profiles
def calculate_customer_profile(data, product_name):
    avg_age = data['Age'].mean()
    avg_usage = data['Usage'].mean()
    avg_fitness = data['Fitness'].mean()
    avg_income = data['Income'].mean()
    avg_miles = data['Miles'].mean()
    customer_count = len(data)

    return {
        'Product': product_name,
        'Avg_Age': avg_age,
        'Avg_Usage': avg_usage,
        'Avg_Fitness': avg_fitness,
        'Avg_Income': avg_income,
        'Avg_Miles': avg_miles,
        'Customer_Count': customer_count
    }

# Calculate customer profiles for each product
kp281_profile = calculate_customer_profile(kp281_data, 'KP281')
kp481_profile = calculate_customer_profile(kp481_data, 'KP481')
kp781_profile = calculate_customer_profile(kp781_data, 'KP781')

# Create a DataFrame with the profiles
product_profiles = pd.DataFrame([kp281_profile, kp481_profile, kp781_profile])

print("\nCustomer Profiles for Each Product:")
print(product_profiles)
```

```
Customer Profiles for Each Product:
  Product  Avg_Age  Avg_Usage  Avg_Fitness  Avg_Income   Avg_Miles  \
0   KP281    28.55   3.087500       2.9625   46418.025   82.787500
1   KP481    28.90   3.066667       2.9000   48973.650   87.933333
2   KP781    29.10   4.775000       4.6250   75441.575  166.900000

   Customer_Count
0              80
1              60
2              40
```

Figure 3.31: Customer Profiles for Each Product

7.3 Descriptive Analytics - Customer Profile for Age

In [26]:
```
plt.figure(figsize=(8, 6))
ax = sns.boxplot(x='Product', y='Age', data=df)
plt.title('Age Distribution by Product')

# Add annotations for median values
for product in df['Product'].unique():
    y_values = df[df['Product'] == product]['Age']
    median_val = y_values.median()

    plt.annotate(f'Median: {median_val:.2f}',
                 xy=(df['Product'].unique().tolist().index(product), median_val),
                 xytext=(0, 5),
                 textcoords='offset points',
                 ha='center', va='bottom')
plt.show()
```

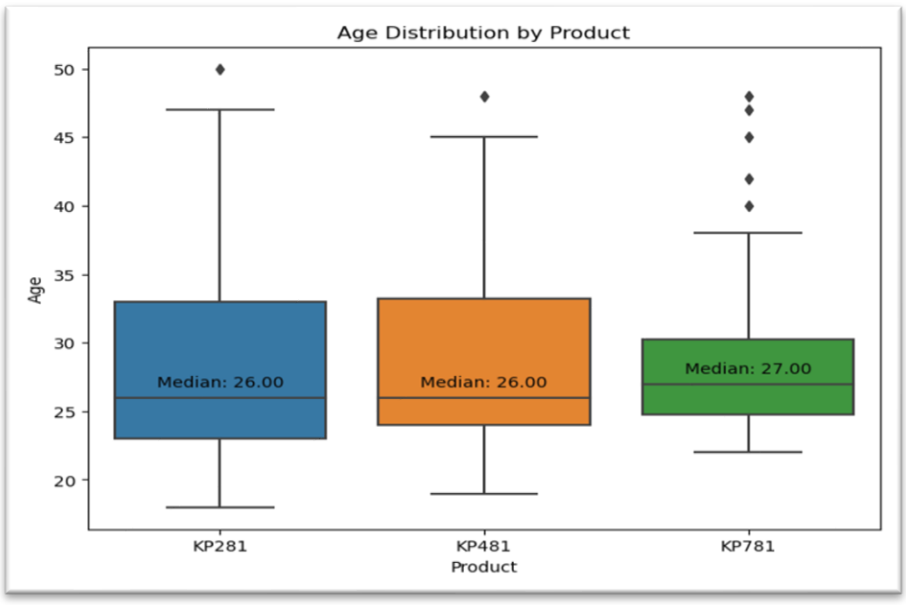


Figure 3.32: Descriptive Analytics - Customer Profile for Age

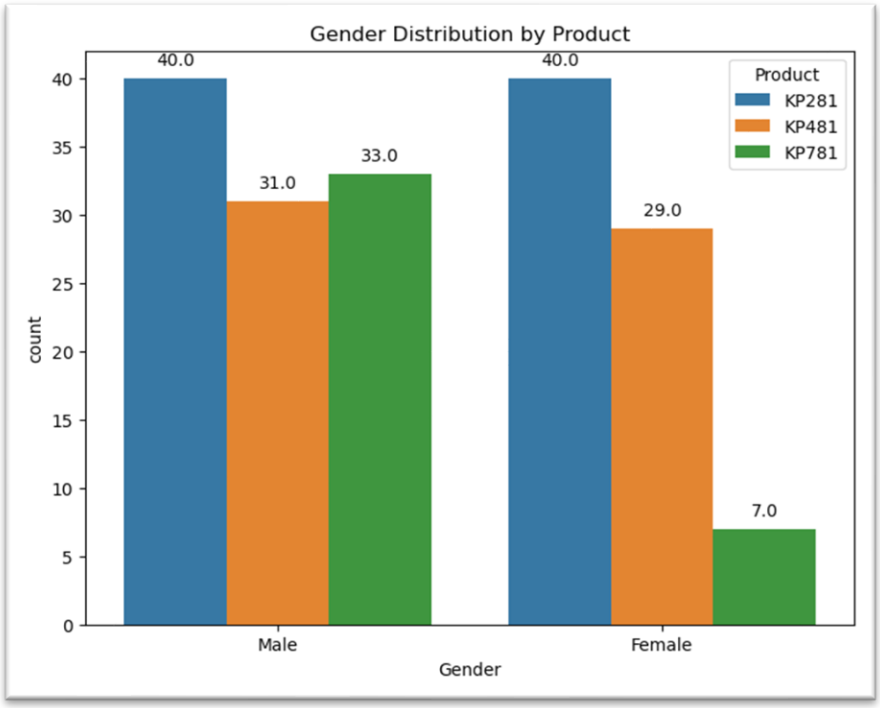


Figure 3.33: Descriptive Analytics - Customer Profile for Gender

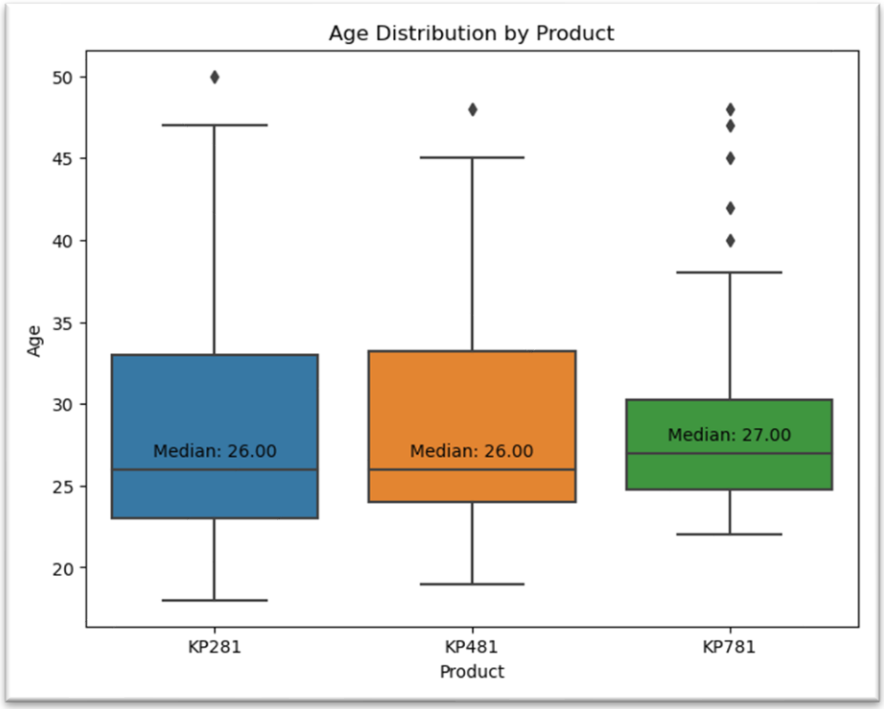


Figure 3.34: Descriptive Analytics - Customer Profile for Age

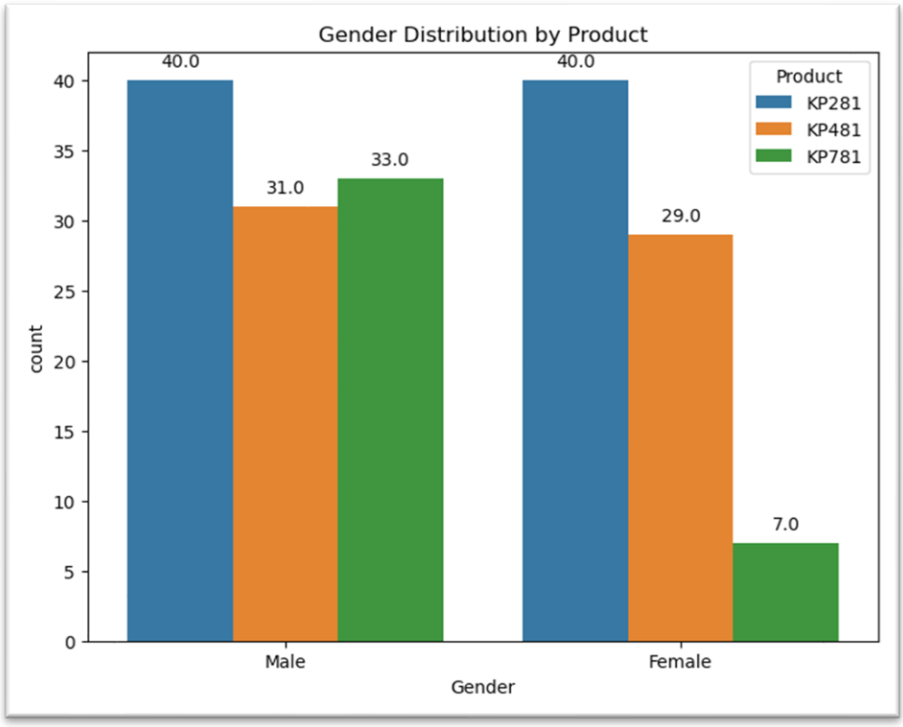


Figure 3.35: Descriptive Analytics – Gender distribution by product

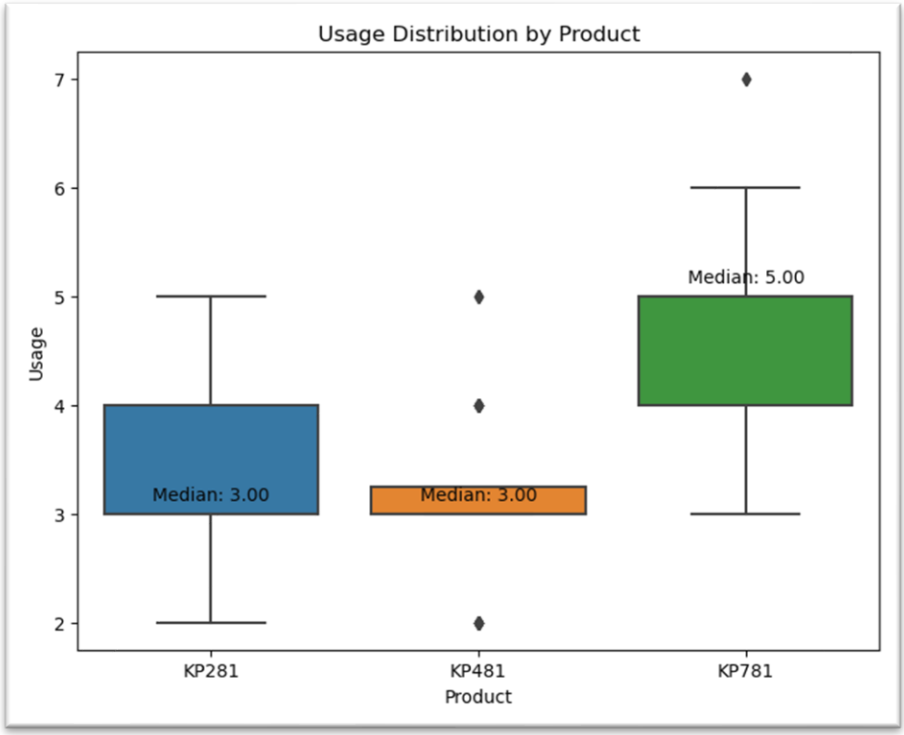


Figure 3.36: Descriptive Analytics – Usage distribution by product

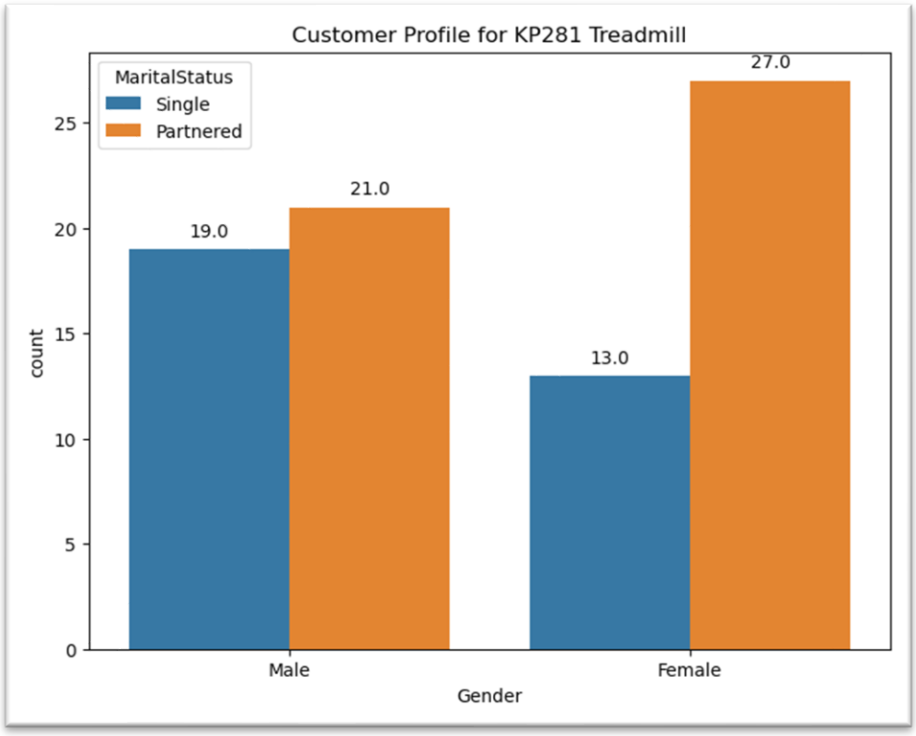


Figure 3.37: Descriptive Analytics – Customer profile for KP281 treadmill

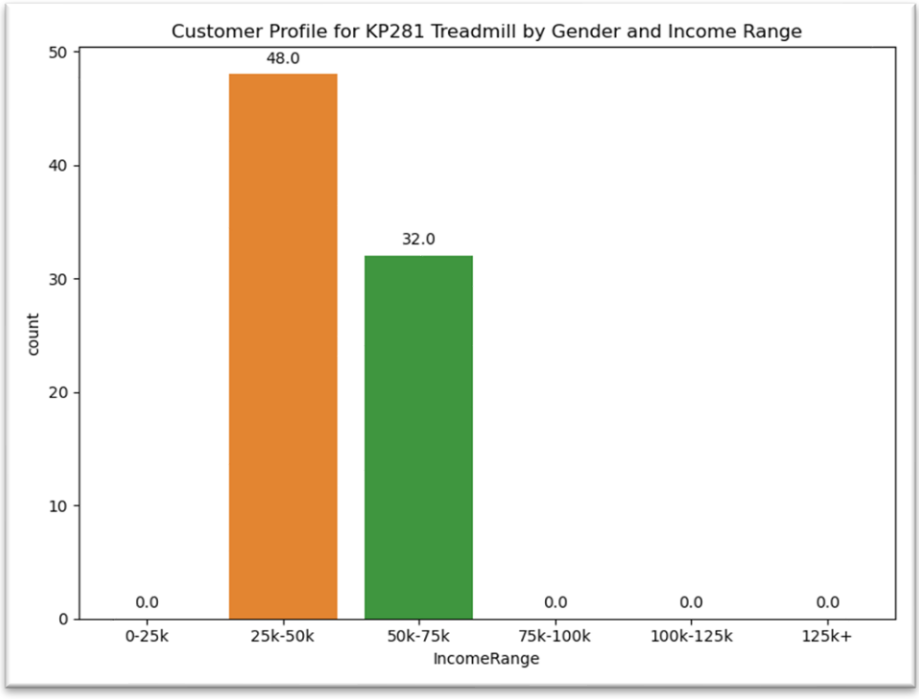


Figure 3.38: Descriptive Analytics – Customer profile for KP281 treadmill by gender and income range

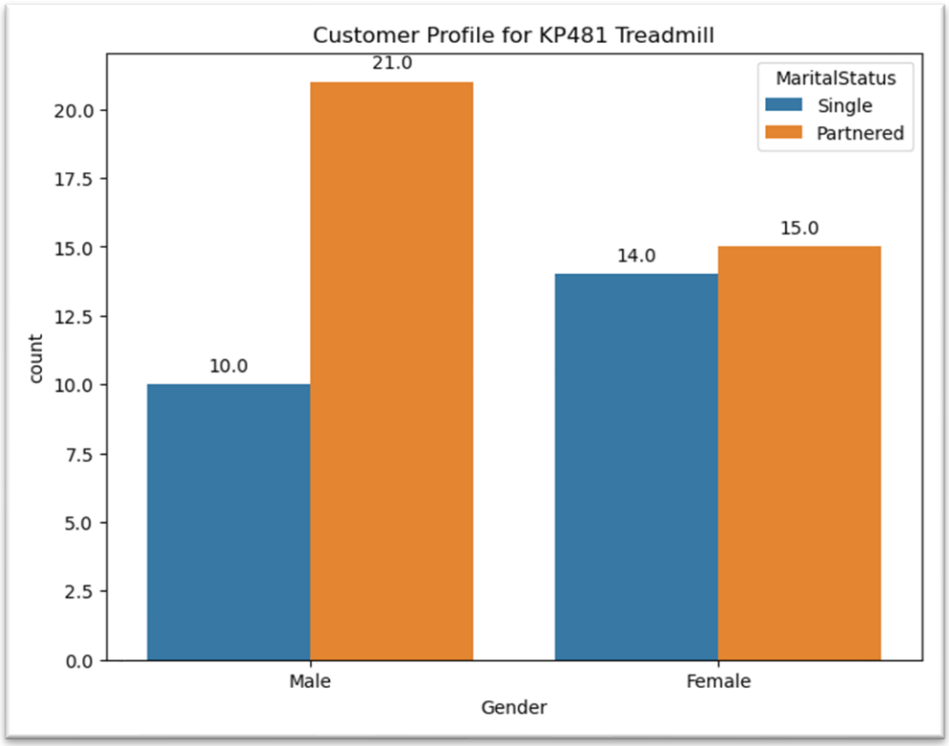


Figure 3.39: Descriptive Analytics – Customer profile for KP481 treadmill

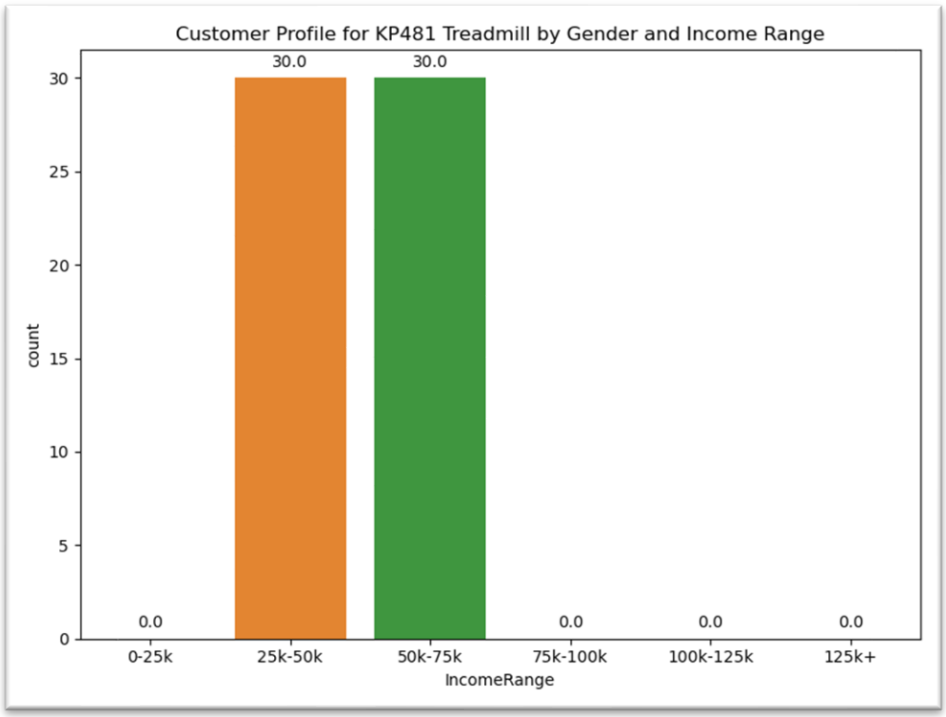


Figure 3.40: Descriptive Analytics – Customer profile for KP481 treadmill by gender and income range

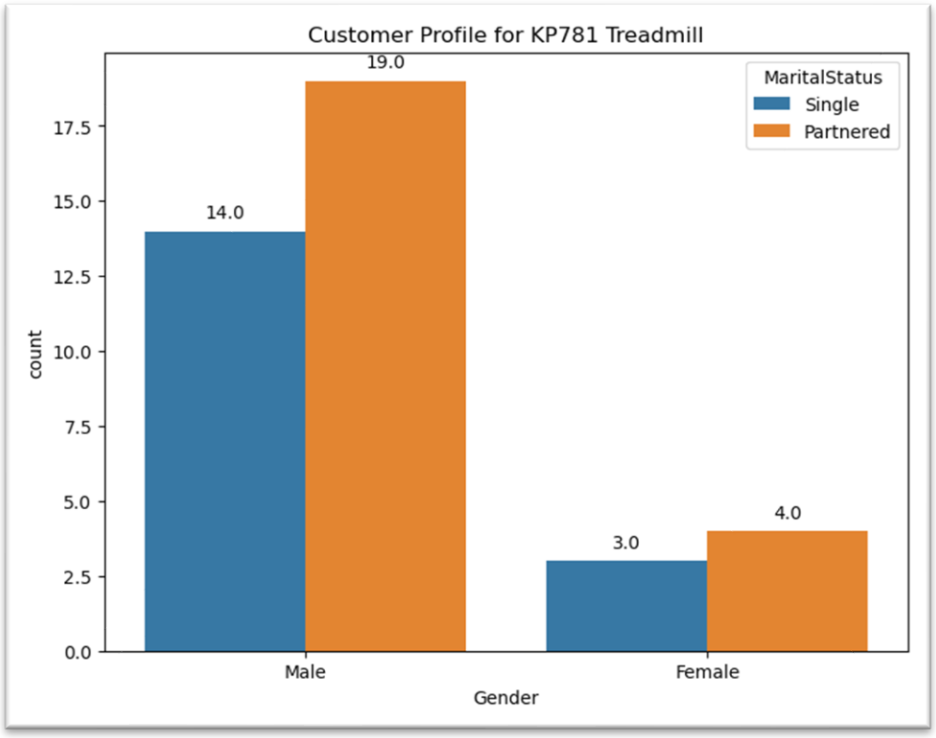


Figure 3.41: Descriptive Analytics – Customer profile for KP781 treadmill

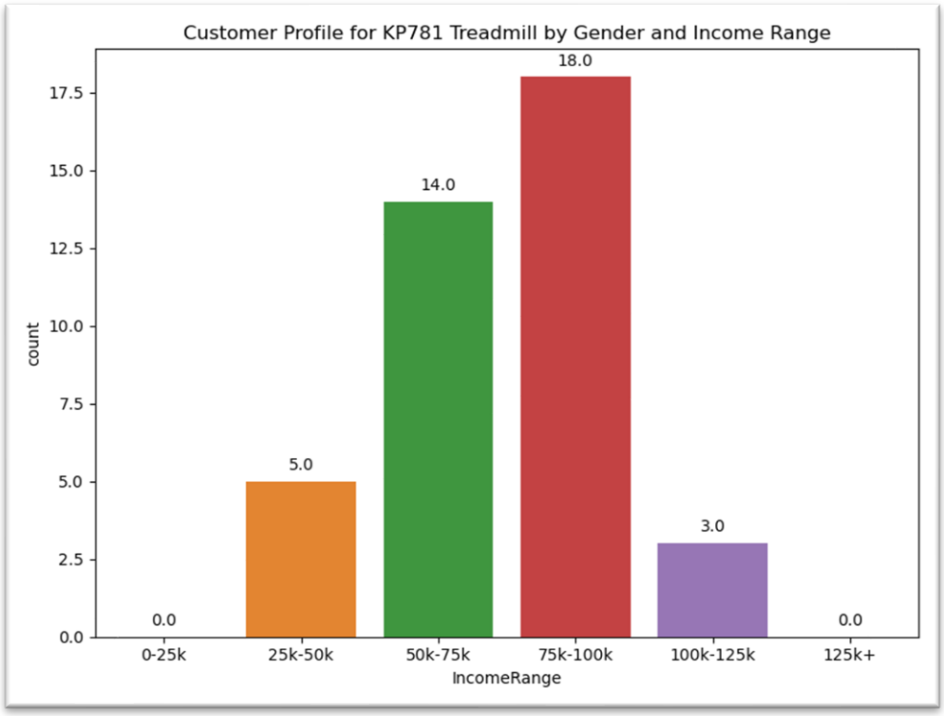


Figure 3.42: Descriptive Analytics – Customer profile for KP781 treadmill by gender and income range

Recommendations :

1. Customer Engagement:	1. We should focus on age-specific marketing campaigns to target and engage customers in the 20-35 age range, as this group shows the highest interest in all our treadmills. 2. Most of our KP281 and KP481 treadmill customers fall into the 20-35 age range whereas our KP781 treadmill customers fall into the 25-30 age range. 3. The average age of our customers is 28.79 years, and median age is 26.0 years. 4. We should also focus on marketing campaigns to target and engage married customers as they are more than single customers. 5. For our KP781, we have very fewer female customers, so we can introduce some deals or discounts so that more females who are athletes but our KP781.
2. Focus on each treadmill model:	1. We must focus on treadmill-model-specific marketing campaigns to target and engage customers of each model. 2. 44.44% of our customers are KP281 buyers, 33.33% are KP481 buyers and 22.22% are KP781 buyers. 3. Whereas generally customers of KP781 (which is our highest priced product) have more than 50,000 dollars annual income and are mostly active fitness enthusiasts or athletes. 4. So, we have a balanced customer base for each treadmill model, and therefore marketing and attention to each treadmill model is equally important.

3. Promote Fitness Features:	1. We must consider developing and promoting treadmill features that cater to customers with fitness levels of 3 or more, as they represent a significant portion of the market. 2. Most of our consumers have moderate to high levels of fitness. 3. For our KP781 model, buyers are mostly athletes or active fitness enthusiasts, so we can promote our KP781 model's features more to such customers so that its sales can increase.
4. Advertise Usage Scenarios:	1. We can create promotional materials highlighting the durability and features suitable for customers planning to use treadmills intensively (Usage level 3 or more). 2. Most of our consumers intend to use treadmills 3 times a week or more.
5. Affordability Awareness:	1. We need to leverage messaging that emphasizes affordability to appeal to customers with incomes of 50,000 dollars or more, ensuring they are aware of the budget-friendly options available. 2. Most of our KP281 and KP481 treadmill consumers fall in 40,000 - 60,000 dollars annual income bracket, but our KP781 treadmill customers fall in 60,000 - 90,000 dollars annual income bracket. 3. We note that most customers buying KP281 fall under 25,000 - 50,000 dollars annual income bracket. So, if we offer them some discounts or offers, they can shift to KP481. 4. We note that 50% of customers buying KP481 fall under 50,000 - 75,000 dollars annual income bracket. So, if we offer them some discounts or offers, they can shift to KP781.

6. Inclusive Marketing:	1. We must ensure that our marketing materials represent and speak to both genders, acknowledging the fact that we have a diverse customer base in terms of gender. 2. We can focus more on sales to female customers because they are little less in number than male customers.
7. Personalized Recommendations:	1. We can implement a recommendation system or personalized customer outreach to suggest treadmills based on individual preferences and needs.

Table 3.1: Recommendations

Case Study Link:

https://github.com/MaharshiDSML/Aerofit-Company-Case-Study-Descriptive-Statistics-Probability

Chapter 4: Business Case Study 4- Confidence Interval and CLT

Problem Description:

1. **An American multinational retail corporation** that operates a chain of supercenters, discount departmental stores, and grocery stores from the United States.

2. The company has more than 100 million customers worldwide.

3. The Management team wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions.

4. They want to understand if the **spending** habits differ between male and female customers: Do women spend more on Black Friday than men? (Assume 50 million customers are male and 50 million are female).

Business Questions to be answered from Analysis:

1. Are women spending more money per transaction than men? Why or why not?
2. Confidence intervals and distribution of the mean of the expenses by female and male customers.
3. Are confidence intervals of average male and female spending overlapping? How can a company leverage this conclusion to make changes or improvements?
4. Results when the same activity is performed for Married vs Unmarried?
5. Results when the same activity is performed for Age.

Methodology/Approach to Solve the Case Study:

1. We will be defining **problem statement and analyzing basic metrics:**
    1. Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), statistical summary
    2. Non-Graphical Analysis: Value counts and unique attributes
    3. Visual Analysis - Univariate & Bivariate:
        1. For continuous variable(s): Distplot, Countplot, Histogram for univariate analysis.
        2. For categorical variable(s): Boxplot.
        3. For correlation: Heatmaps, Pairplots.
2. We will perform **missing value & outlier detection.**
3. We will draw business insights based on **non- graphical and visual analysis:**
    1. Comments on the range of attributes.
    2. Comments on the distribution of the variables and relationship between them.
    3. Comments for each univariate and bivariate plot.
4. We will draw **final insights:**
    1. Illustrate the insights based on exploration and CLT.
    2. Comments on the distribution of the variables and relationship between them.
    3. Comments for each univariate and bivariate plots.
    4. Comments on different variables when generalizing it for Population.
5. We will make **recommendations:**
    1. Actionable items for business. No technical jargon. No complications. Simple action items that everyone can understand.

Analysis:

1. Defining Problem Statement and Analyzing basic metrics:

1A. Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required) and statistical summary.

In [1]:
```
# Importing the libraries:

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import warnings  # Ignoring future Jupyter warnings:
warnings.simplefilter(action = 'ignore', category = FutureWarning)
```

In [2]:
```
# Reading the Walmart data:

df = pd.read_csv('walmart_data.csv')
df
```

Out[2]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category	Purchase
0	1000001	P00069042	F	0-17	10	A	2	0	3	8370
1	1000001	P00248942	F	0-17	10	A	2	0	1	15200
2	1000001	P00087842	F	0-17	10	A	2	0	12	1422
3	1000001	P00085442	F	0-17	10	A	2	0	12	1057
4	1000002	P00285442	M	55+	16	C	4+	0	8	7969
...	...	...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	1	20	368
550064	1006035	P00375436	F	26-35	1	C	3	0	20	371
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	137
550066	1006038	P00375436	F	55+	1	C	2	0	20	365
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	490

550068 rows × 10 columns

Figure 4.1: Defining Problem Statement and Analyzing basic metrics

```
In [6]: # Checking the general information and datatypes of the data:

        df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 550068 entries, 0 to 550067
        Data columns (total 10 columns):
         #   Column                      Non-Null Count   Dtype
        ---  ------                      --------------   -----
         0   User_ID                     550068 non-null  int64
         1   Product_ID                  550068 non-null  object
         2   Gender                      550068 non-null  object
         3   Age                         550068 non-null  object
         4   Occupation                  550068 non-null  int64
         5   City_Category               550068 non-null  object
         6   Stay_In_Current_City_Years  550068 non-null  object
         7   Marital_Status              550068 non-null  int64
         8   Product_Category            550068 non-null  int64
         9   Purchase                    550068 non-null  int64
        dtypes: int64(5), object(5)
        memory usage: 42.0+ MB

In [7]: # Checking the structure of the dataframe using "shape attribute":

        df.shape

Out[7]: (550068, 10)
```

Figure 4.2: Checking the general information, datatypes and shape of the data

```
In [8]: # Conversion of categorical attributes to "categorical variables":

        cols = ['User_ID', 'Occupation', 'Marital_Status', 'Product_Category']
        df[cols] = df[cols].astype('object')
        df.dtypes

Out[8]: User_ID                     object
        Product_ID                  object
        Gender                      object
        Age                         object
        Occupation                  object
        City_Category               object
        Stay_In_Current_City_Years  object
        Marital_Status              object
        Product_Category            object
        Purchase                     int64
        dtype: object

In [9]: # Checking the general information and datatypes of the data again to verify "newly formed categorical variables":

        df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 550068 entries, 0 to 550067
        Data columns (total 10 columns):
         #   Column                      Non-Null Count   Dtype
        ---  ------                      --------------   -----
         0   User_ID                     550068 non-null  object
         1   Product_ID                  550068 non-null  object
         2   Gender                      550068 non-null  object
         3   Age                         550068 non-null  object
         4   Occupation                  550068 non-null  object
         5   City_Category               550068 non-null  object
         6   Stay_In_Current_City_Years  550068 non-null  object
         7   Marital_Status              550068 non-null  object
         8   Product_Category            550068 non-null  object
         9   Purchase                    550068 non-null  int64
        dtypes: int64(1), object(9)
        memory usage: 42.0+ MB
```

Figure 4.3: Conversion of categorical attributes to "categorical variables"

```
In [10]: # Checking the characteristics and statistical information of the data:
         df.describe()
```

Out[10]:

	Purchase
count	550068.000000
mean	9263.968713
std	5023.065394
min	12.000000
25%	5823.000000
50%	8047.000000
75%	12054.000000
max	23961.000000

```
In [11]: # Checking the characteristics and statistical information of the categorical variable's data:
         df.describe(include = 'object')
```

Out[11]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category
count	550068	550068	550068	550068	550068	550068	550068	550068	550068
unique	5891	3631	2	7	21	3	5	2	20
top	1001680	P00265242	M	26-35	4	B	1	0	5
freq	1026	1880	414259	219587	72308	231173	193821	324731	150933

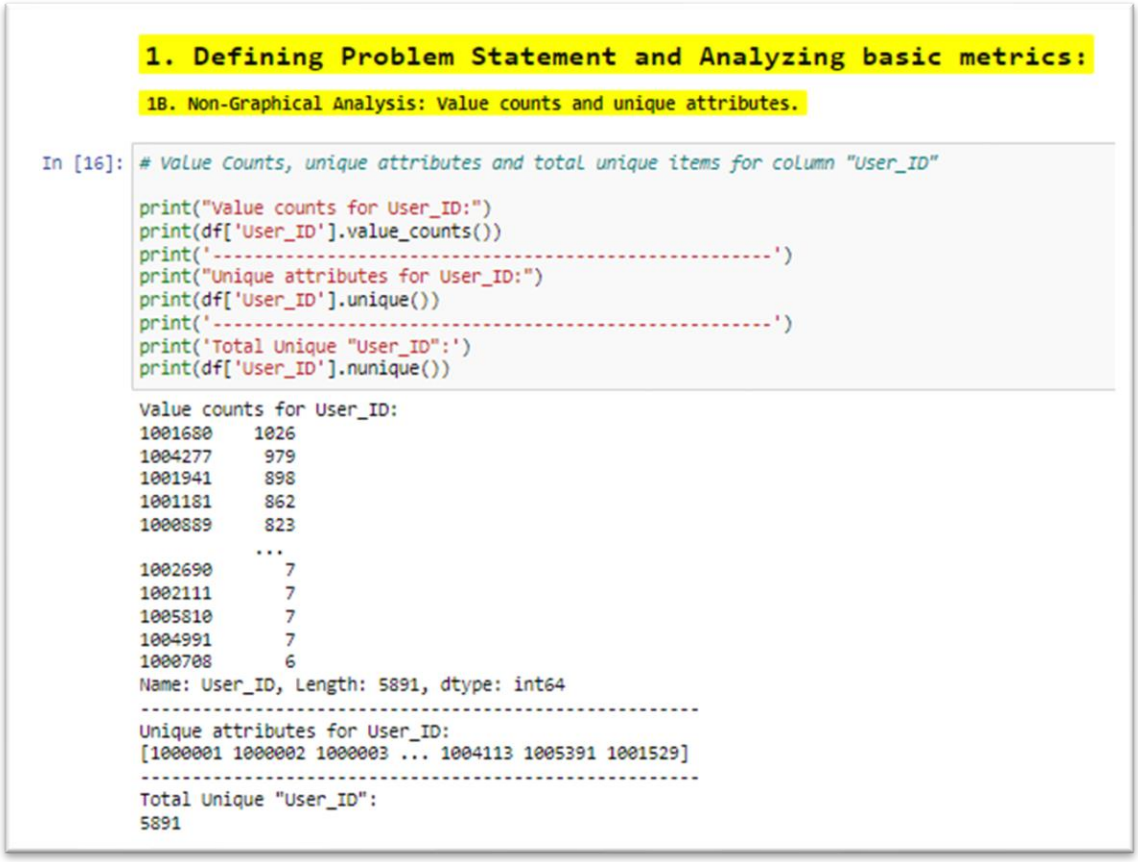
Figure 4.4: Checking the characteristics and statistical information of the data

Inferences:

1. There are no missing values in the data.
2. There are 3631 unique product IDs in the dataset.
3. P00265242 is the best-selling Product ID.
4. There are 7 unique age groups and most of the purchase belongs to age 26-35 group.
5. There are 3 unique city categories with category B being the highest.
6. 5 unique values for Stay_in_current_city_years with 1 being the highest.
7. The difference between mean and median seems to be significant for purchase that suggests outliers in the data.
8. Minimum & Maximum purchase is 12 and 23961 which suggests the purchasing behavior is quite spread over a significant range of values.
9. Mean is 9264 and 75% of purchase is of less than or equal to 12054. It suggest most of the purchase is not more than 12k.
10. Few categorical variables are of integer data type. It can be converted to character type, which we have done.

11. Out of 550068 data points, 414259's gender is Male, and rest are the female.

12. Male purchase count is much higher than female.
13. Standard deviation for purchase has significant value which suggests data is more spread out for this attribute.
14. There are 5891 unique users, and userid 1001680 with the highest count.
15. The customers belong to 21 distinct occupations for the purchases being made with Occupation 4 being the highest.
16. Unmarried people contribute more in terms of the count for the purchase.
17. There are 20 unique product categories with 5 being the highest.



1. Defining Problem Statement and Analyzing basic metrics:

1B. Non-Graphical Analysis: Value counts and unique attributes.

In [16]:
```
# Value Counts, unique attributes and total unique items for column "User_ID"

print("Value counts for User_ID:")
print(df['User_ID'].value_counts())
print('-------------------------------------------------------')
print("Unique attributes for User_ID:")
print(df['User_ID'].unique())
print('-------------------------------------------------------')
print('Total Unique "User_ID":')
print(df['User_ID'].nunique())
```

```
Value counts for User_ID:
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
-------------------------------------------------------
Unique attributes for User_ID:
[1000001 1000002 1000003 ... 1004113 1005391 1001529]
-------------------------------------------------------
Total Unique "User_ID":
5891
```

Figure 4.5: Non-graphical analysis- getting value counts and unique attributes for all columns

- **Similarly getting value counts and unique attributes for all columns.**

Inferences:

1. Approximately 80% of the users are between the age 18-50 (40%: 26-35, 18%: 18-25, 20%: 36-45).
2. 75% of the users are Male and 25% are Female.

3. 60% are Single and 40% are Married.
4. 35% stay in the city from 1 year, 18% from 2 years, 17% from 3 years.
5. A total of 20 product categories are there.
6. There are 5891 different types of User_ID, 3631 different types of Product_ID, 20 different types of occupations, 3 different types of City_Category, 5 different types of Stay_In_Current_City_Years and 20 different types of Product Categories in the city.
7. We can see 35% of the users are aged 26-35. 73% of users are aged between 18-45.
8. From the "Age" observation we saw 40% of the purchase are made by users aged 26-35. And, we have 35% of users aged between 26-35 and they are contributing 40% of total purchase count. So, we can infer users aged 26-35 are more frequent customers.
9. We have 72% male users and 28% female users. Combining with previous observations we can see 72% of male users contributing to 75% of the purchase count and 28% of female users are contributing to 25% of the purchase count.
10. 53% of the users belong to city category C whereas 29% to category B and 18% belong to category A. Combining from the previous observation category B purchase count is 42% and Category C purchase count is 31%. We can clearly see category B are more actively purchasing in spite of the fact they are only 28% of the total users. On the other hand, we have 53% of category C users but they only contribute 31% of the total purchase count.
11. We have seen earlier that city category B and A constitute less percentage of total population, but they contribute more towards purchase count. We can see from the above results large percentage of customers aged 26-35 for B(40%) and A (50%) which can be the reason for these city categories to be more actively purchasing.

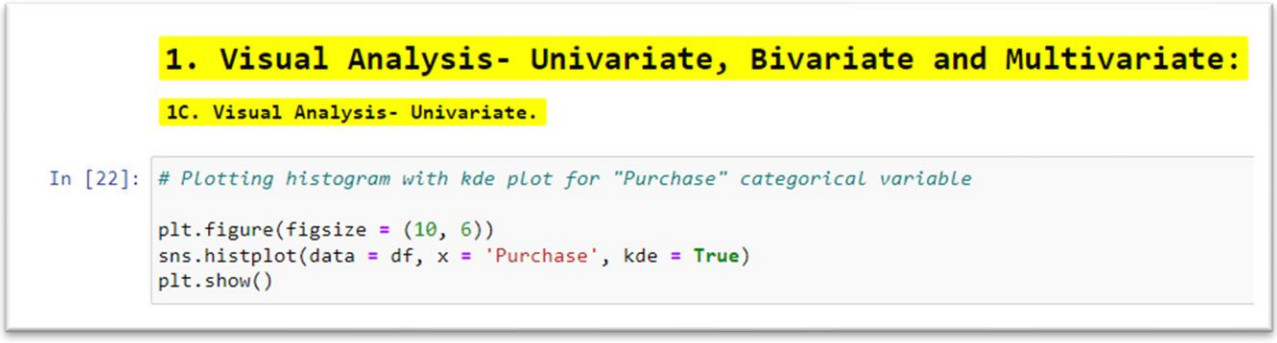


Figure 4.6: Visual Analysis- Univariate

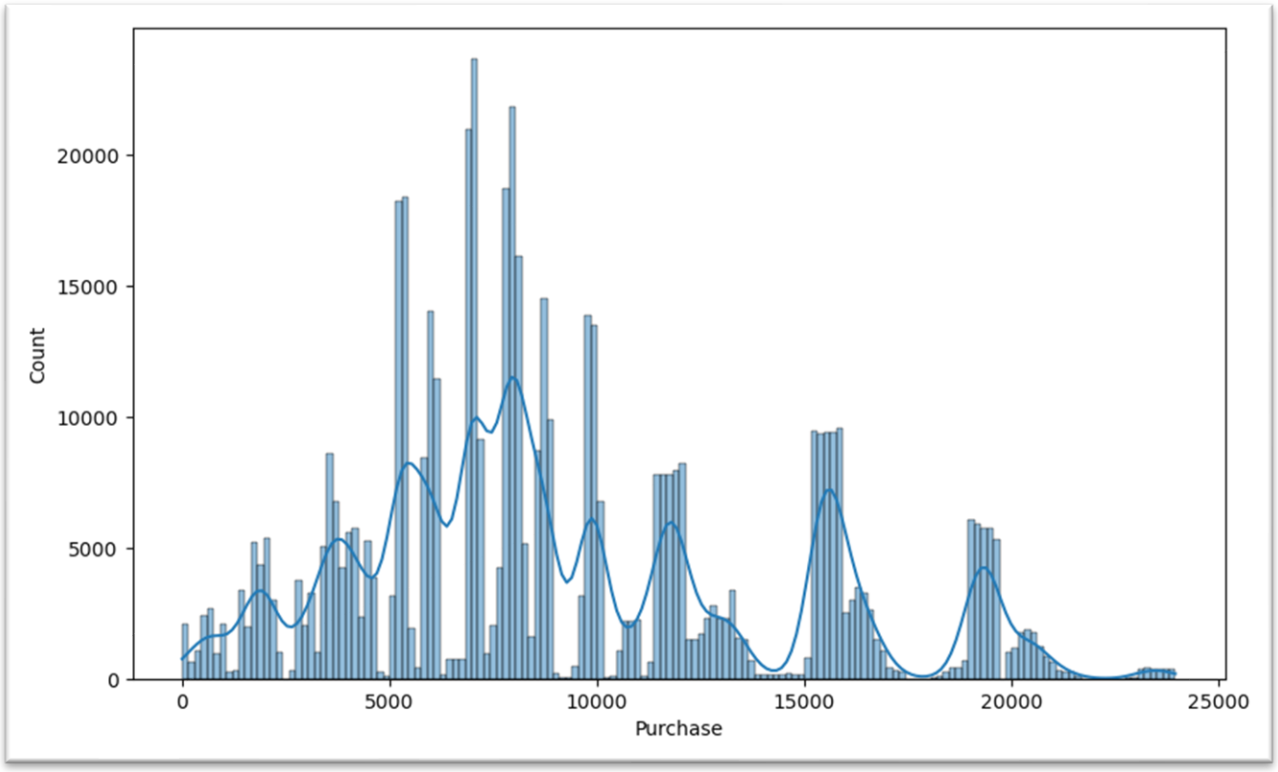


Figure 4.7: Plotting histogram with kde plot for "Purchase" categorical variable

Inferences:

1. We can see purchase value between 5000 and 10000 have higher count. From the initial observation we have already seen the mean and median is 9263 and 8047 respectively. Also, we can see there are outliers in the data.

```
In [23]: # Plotting box plot for "Purchase" categorical variable

         plt.figure(figsize=(5, 4))
         sns.boxplot(data = df, y = b'Purchase')
         plt.show()
```

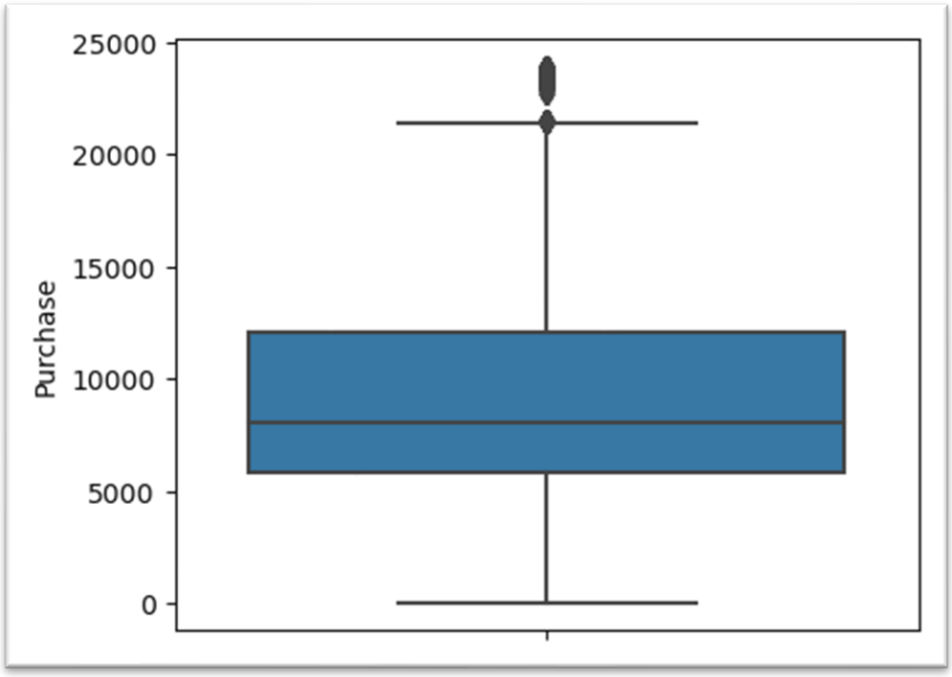


Figure 4.8: Plotting box plot for "Purchase" categorical variable

Inferences:

1. We can see there are outliers in the data for purchase.

```
In [29]: # Plotting histogram with kde plot for "Occupation" categorical variable

         plt.figure(figsize = (7, 3))
         sns.histplot(data = df, x = 'Occupation', kde = True, color = 'orange')
         plt.show()
```

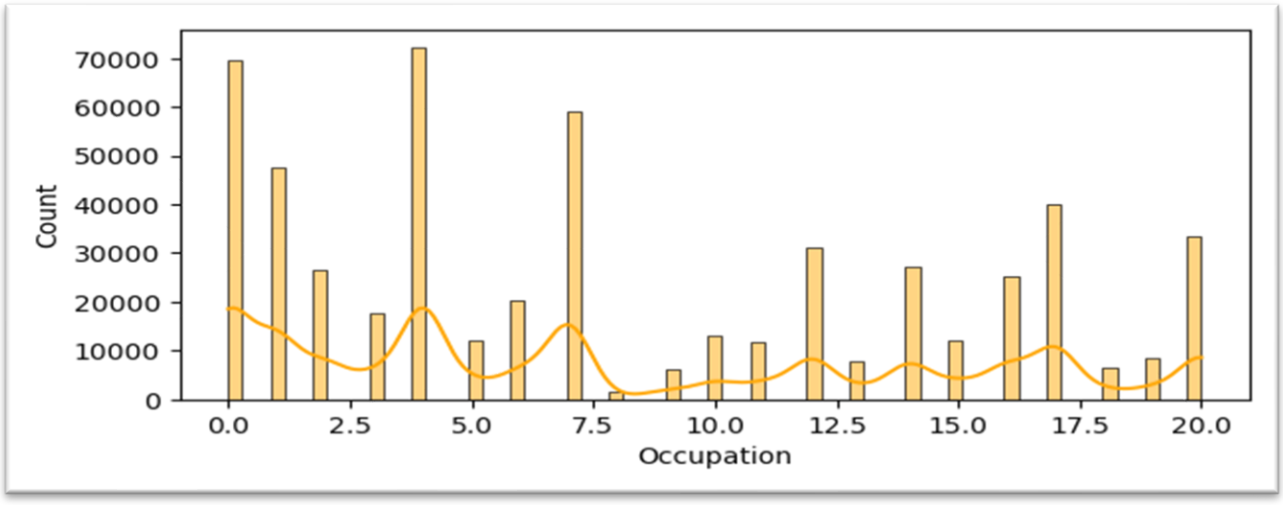
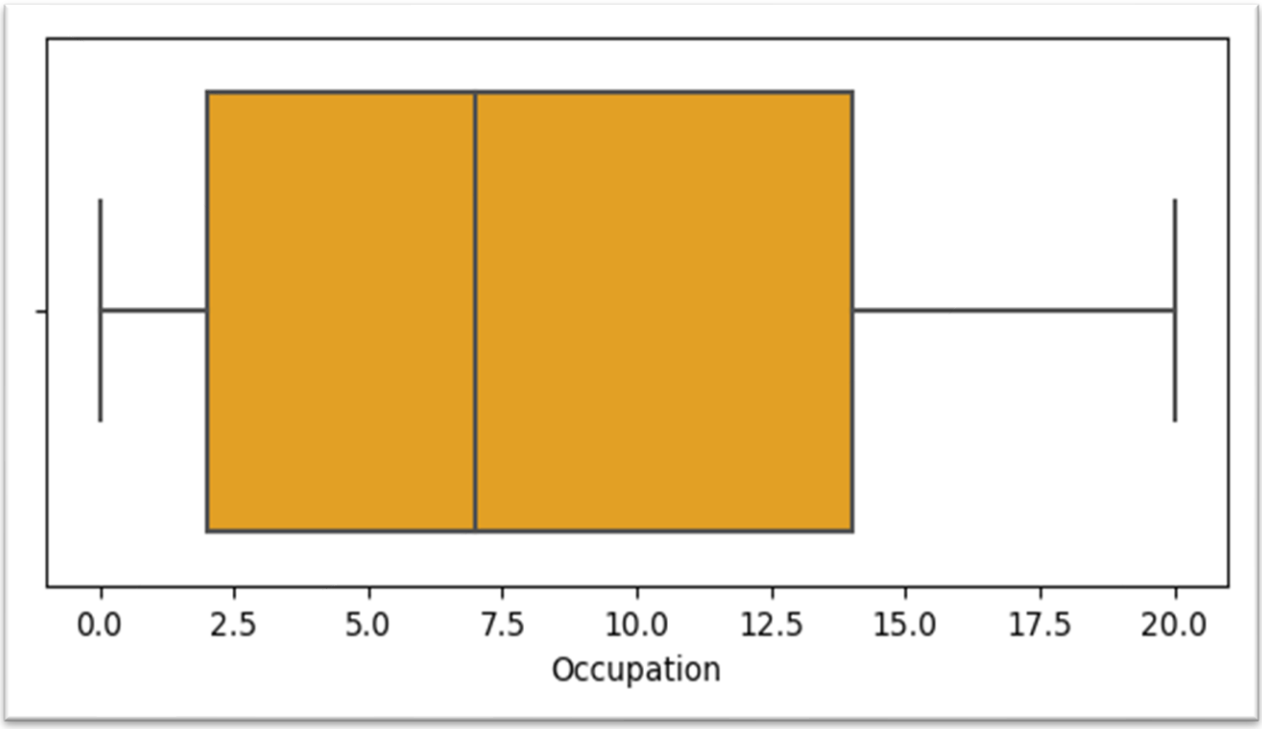
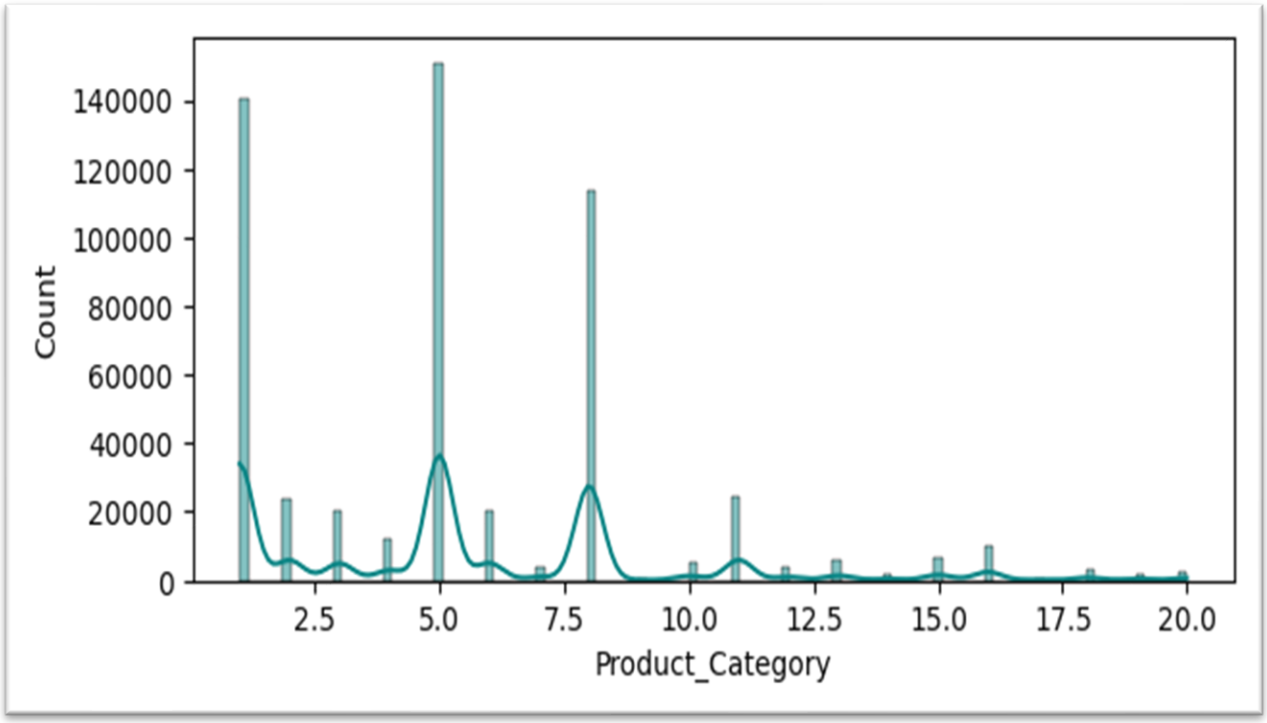
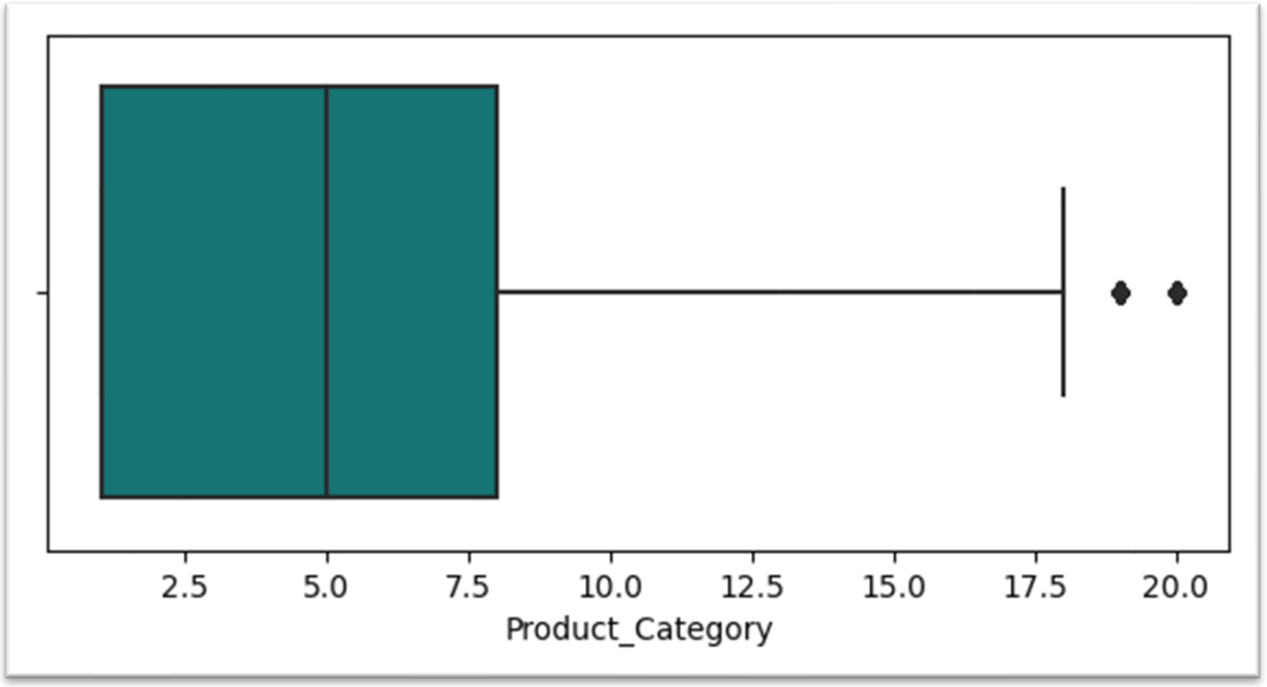


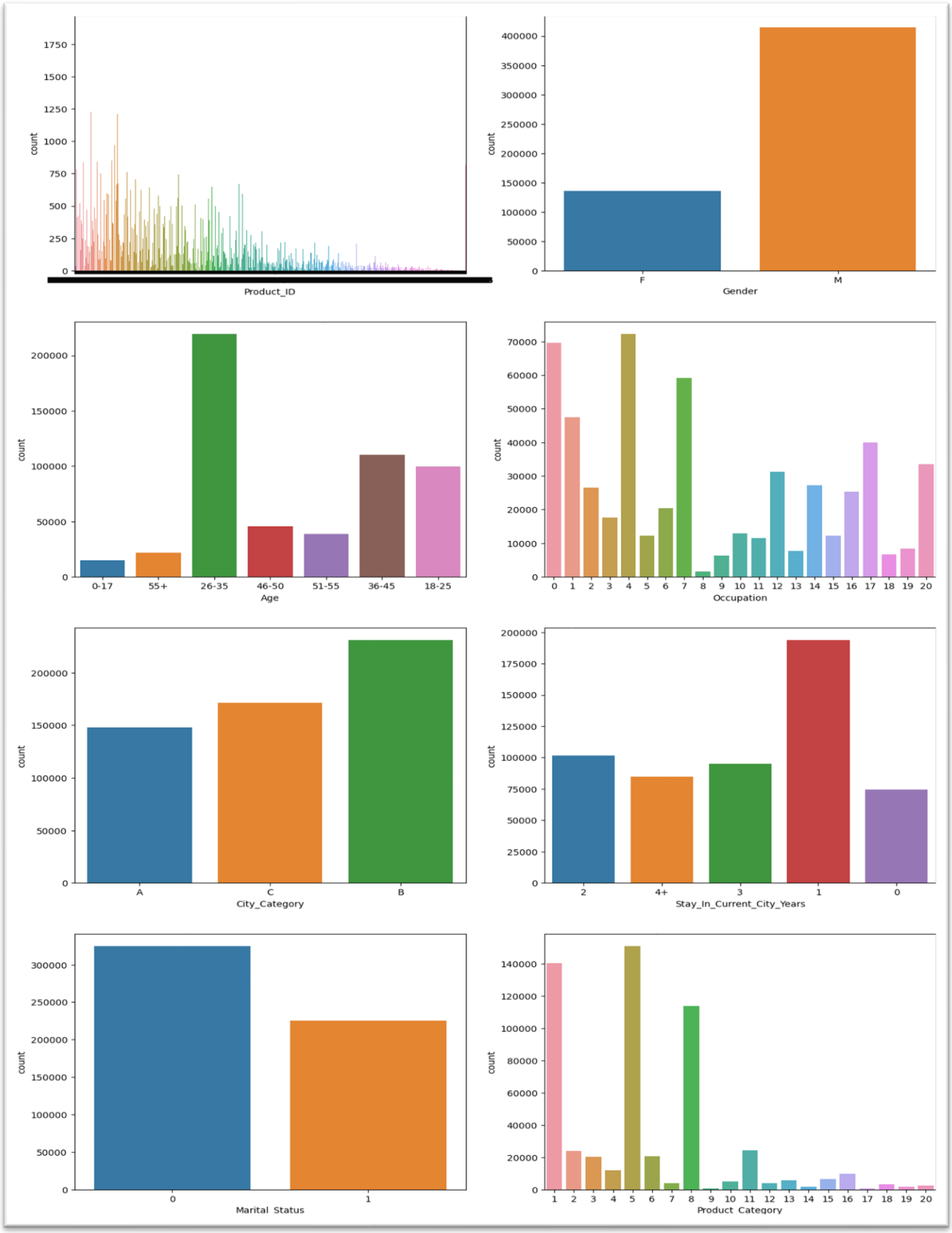
Figure 4.9: Plotting histogram with kde plot for "Occupation" categorical variable

- **Similarly plotting for all categorical variables.**




140000
120000
100000
80000
60000
40000
20000
0
Count
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Product_Category


2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Product_Category


count
Product_ID
count
F
M
Gender
count
0-17
55+
26-35
46-50
51-55
36-45
18-25
Age
count
Occupation
count
A
C
B
City_Category
count
2
4+
3
1
0
Stay_In_Current_City_Years
count
0
1
Marital Status
count
Product Category

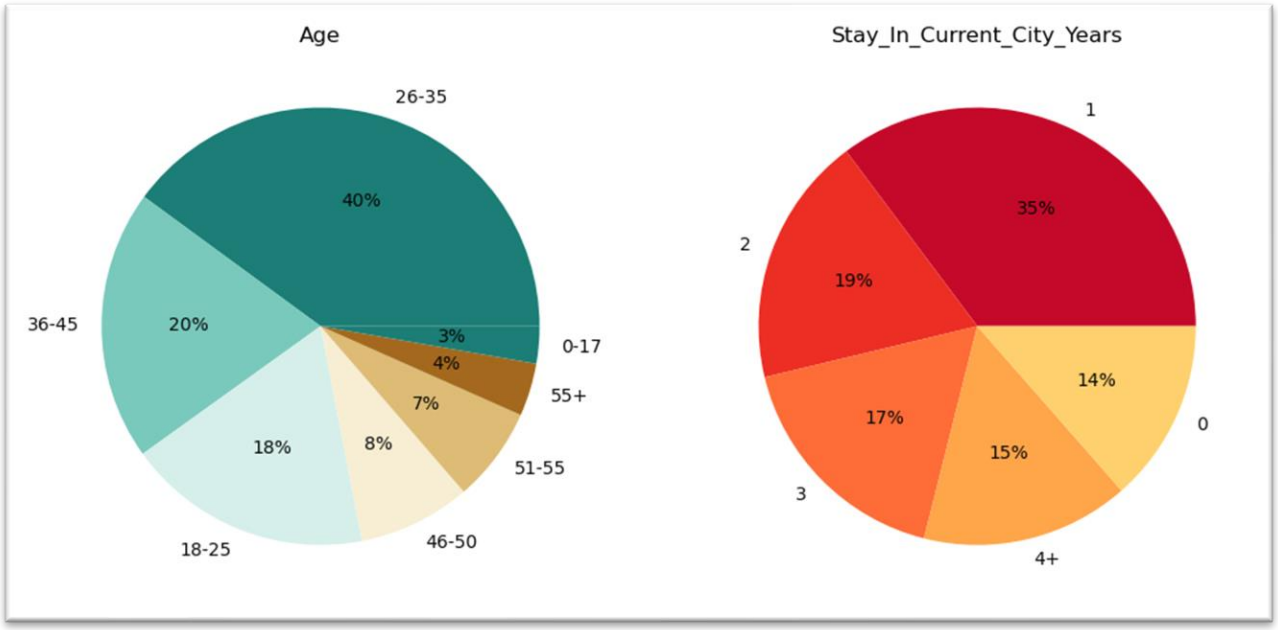


Figure 4.10: Plotting for all other categorical variables

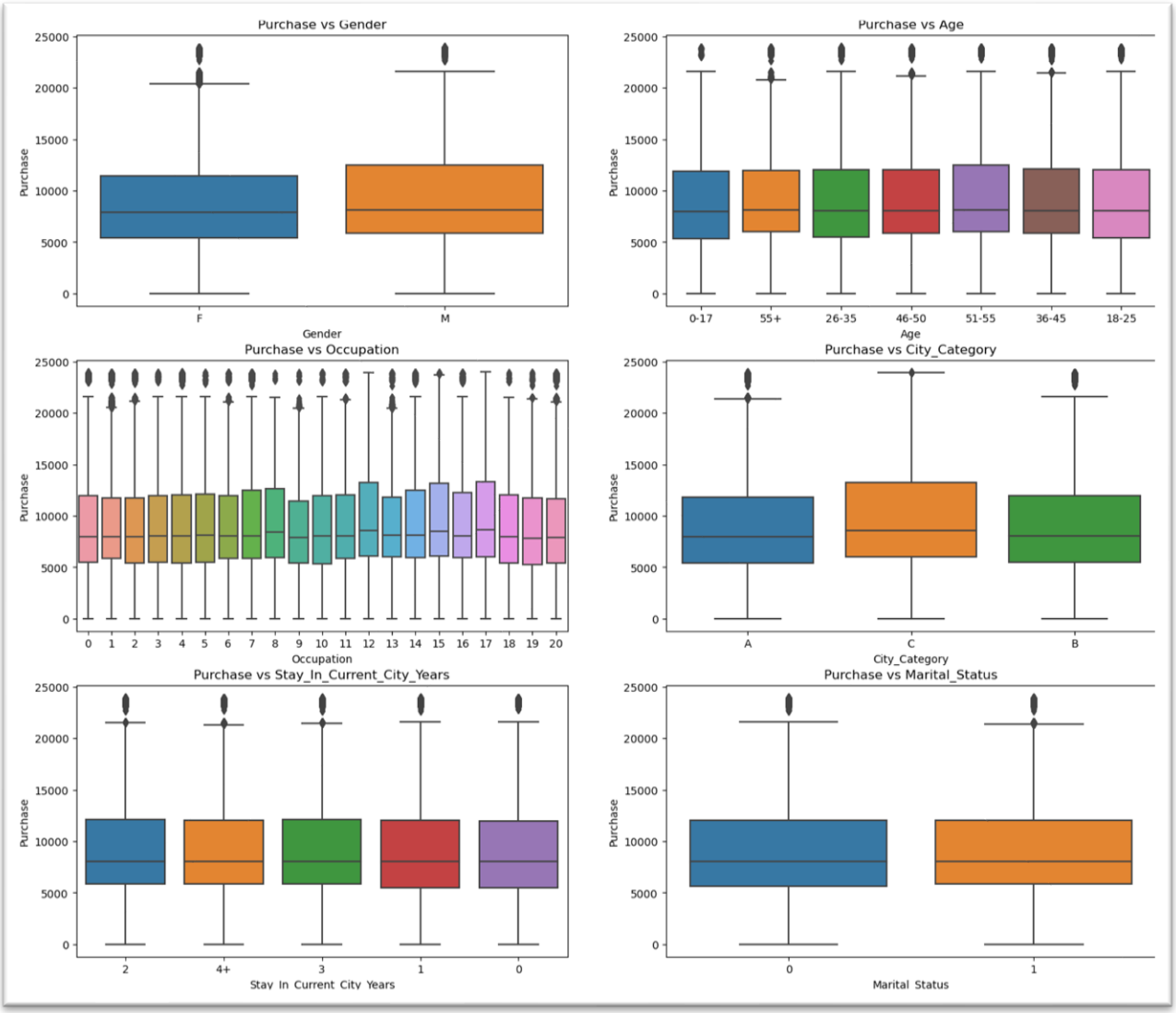
Inferences:

1. We can clearly see from the graphs above the purchases done by males are much higher than females.
2. We have 21 occupations categories. Occupation category 4, 0, and 7 have with higher number of purchases and category 8 with the lowest number of purchase.
3. The purchases are highest from City category B.
4. Single customer purchases are higher than married users.
5. There are 20 product categories with product category 1, 5 and 8 having higher purchasing frequency.
6. Most of the users are Male.
7. There are 21 different types of Occupation and Product_Category.
8. More users belong to B City_Category.
9. More users are Single as compared to Married.
10. Product_Category - 1, 5, 8, & 11 have highest purchasing frequency.

1. Visual Analysis- Univariate, Bivariate and Multivariate:

1C. Visual Analysis- Bivariate.

In [38]:
```
attr = ['Gender', 'Age', 'Occupation', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status']

fig, axs = plt.subplots(nrows = 3, ncols = 2, figsize = (18, 10))
fig.subplots_adjust(top = 1.3)

count = 0
for row in range(3):
    for col in range(2):
        sns.boxplot(data = df, y = 'Purchase', x = attr[count], ax = axs[row, col],)
        axs[row,col].set_title(f"Purchase vs {attr[count]}")
        count += 1

plt.show()

plt.figure(figsize = (8, 5))
sns.boxplot(data = df, y = 'Purchase', x = 'Product_Category')
plt.show()
```



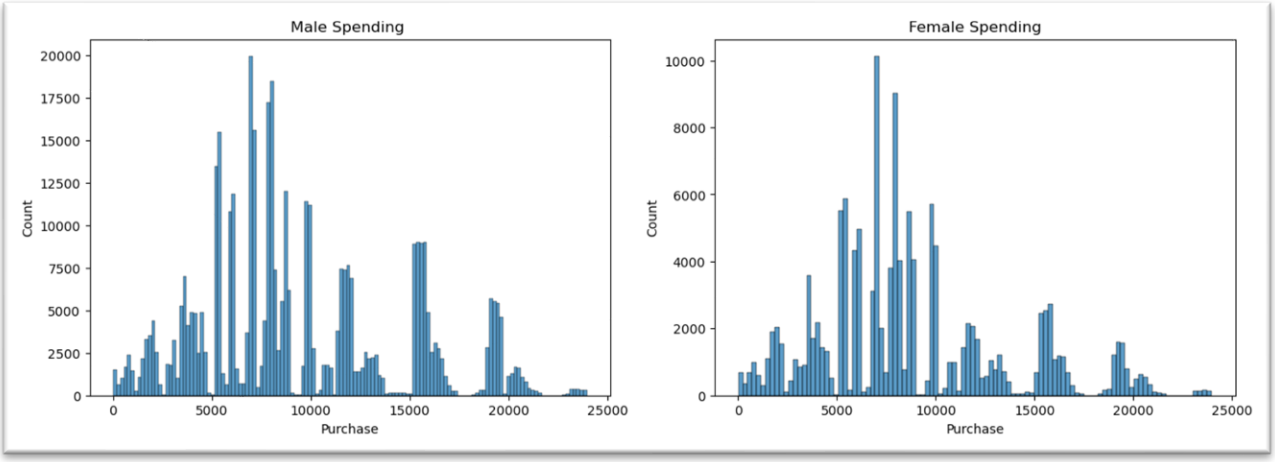


Figure 4.11: Visual Analysis- Bivariate

Inferences:

1. The spending behavior for males and females are similar as we had seen from the above histplots. Males purchasing value are in the little higher range than females.
2. Among different age categories, we see similar purchase behavior. For all age groups, most of the purchases are of values between 5k to 12k with all have some outliers.
3. Among different occupation as well, we see similar purchasing behavior in terms of the purchase values.
4. Similarly for City category, stay in current city years, marital status - we see the users spend mostly in the range of 5k to 12k.
5. We see variations among product categories. Product category 10 products are the costliest ones. Also, there are few outliers for some of the product categories.
6. From the above histplots, we can clearly see spending behavior is very much similar in nature for both males and females as the maximum purchase count are between the purchase value range of 5000-10000 for both. But the purchase count are more in case of males.

2. Visual Analysis- Univariate, Bivariate and Multivariate:

2C. Visual Analysis- Multivariate:

In [39]:
```
fig, axs = plt.subplots(nrows = 2, ncols = 2, figsize = (20, 6))
fig.subplots_adjust(top = 1.5)

sns.boxplot(data = df, y = 'Purchase', x = 'Gender', hue = 'Age', ax = axs[0,0])
sns.boxplot(data = df, y = 'Purchase', x = 'Gender', hue = 'City_Category', ax = axs[0,1])

sns.boxplot(data = df, y = 'Purchase', x = 'Gender', hue = 'Marital_Status', ax = axs[1,0])
sns.boxplot(data = df, y = 'Purchase', x = 'Gender', hue = 'Stay_In_Current_City_Years', ax = axs[1,1])

plt.show()
```

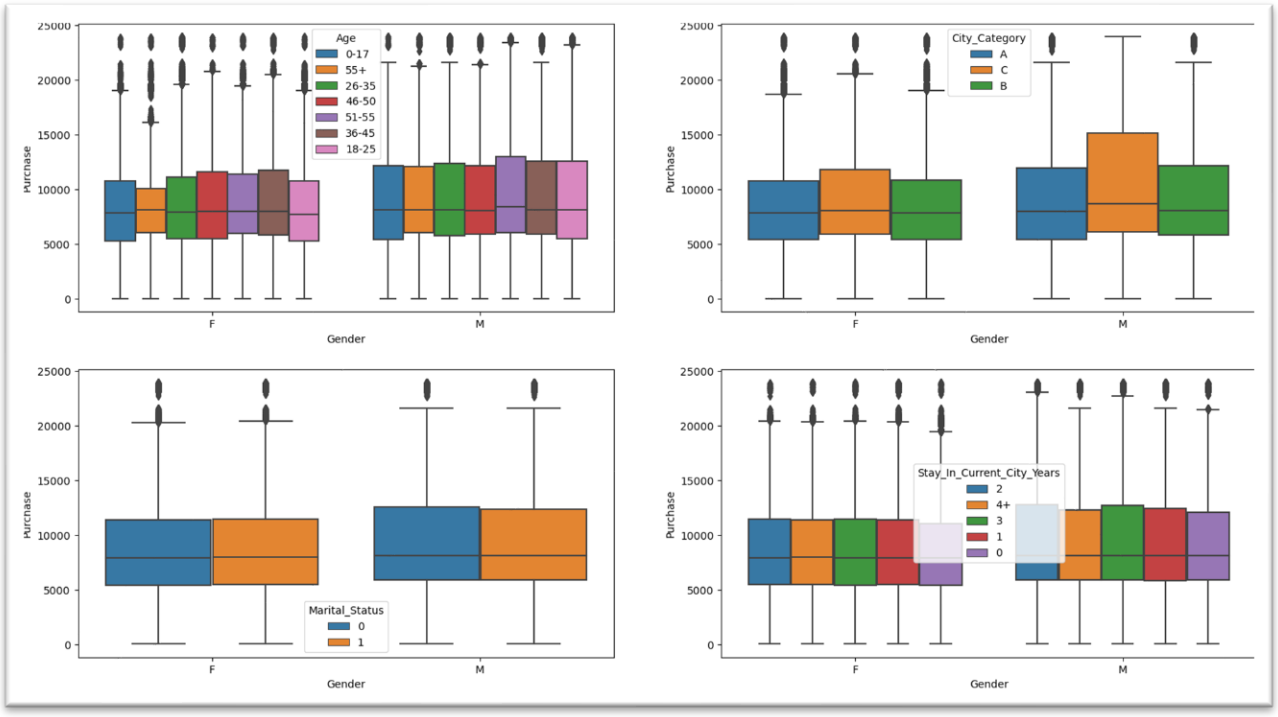


Figure 4.12: Visual Analysis- Multivariate

Inferences:

1. The purchasing pattern is very much similar for males and females even among differnt age groups.
2. The purchasing behavior of males and females on the basis different city categories is also similar in nature. Still, males from city category B tends to purchase costlier products in comparison to females.
3. Males and females spending behavior remains similar even when take into account their marital status.
4. Purchase values are similar for males and females on the basis Stay_in_current_city_years. Although, Males buy slightly high value products.

Correlation between categorical variables:

```
In [40]: sns.heatmap(df.corr(), annot=True, cmap="Blues", linewidth=.5)
Out[40]: <Axes: >
```

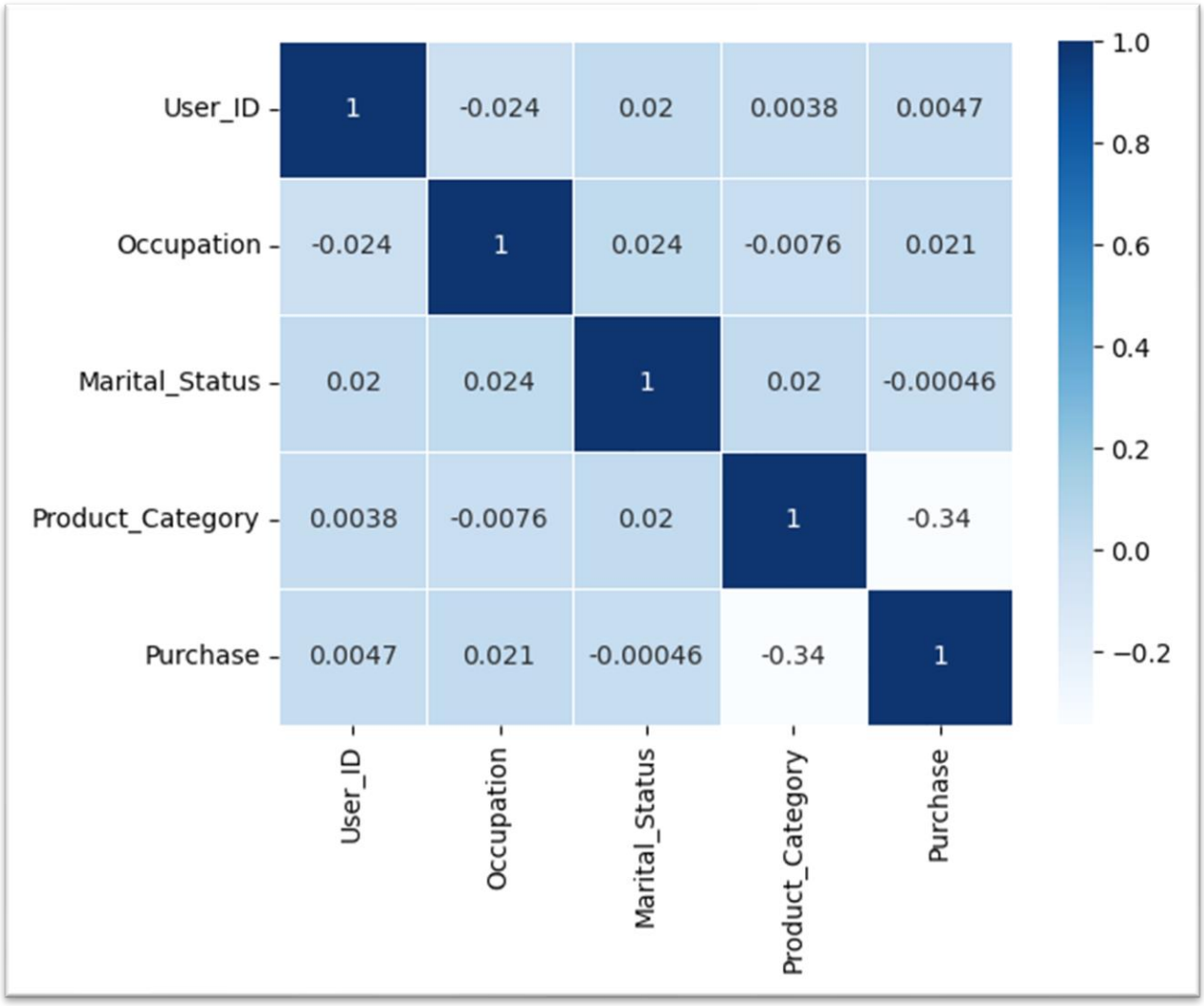


Figure 4.13: Correlation between categorical variables

Inferences:

1. From the above correlation plot, we can see that the correlation is not significant between any pair of variables.

2. Missing Value and Outlier Detection:

2A. Findind missing values.

In [44]:
```
# Checking missing values:

df.isnull().sum()
```

Out[44]:
```
User_ID                     0
Product_ID                  0
Gender                      0
Age                         0
Occupation                  0
City_Category               0
Stay_In_Current_City_Years  0
Marital_Status              0
Product_Category            0
Purchase                    0
dtype: int64
```

Figure 4.14: Finding missing values

Inferences:

1. There are no missing values in the dataset.

2. Missing Value and Outlier Detection:

2B. Finding outliers.

In [45]:
```
# Using pandas describe() to find outliers:

df.describe()
```

Out[45]:

	Purchase
count	550068.000000
mean	9263.968713
std	5023.065394
min	12.000000
25%	5823.000000
50%	8047.000000
75%	12054.000000
max	23961.000000

Figure 4.15: Finding outliers

Inferences:

1. The max Purchase amount is "23961" while its mean is "9263.96".
2. The mean is sensitive to outliers, but the fact the mean is so small compared to the max value indicates the max value is an outlier.

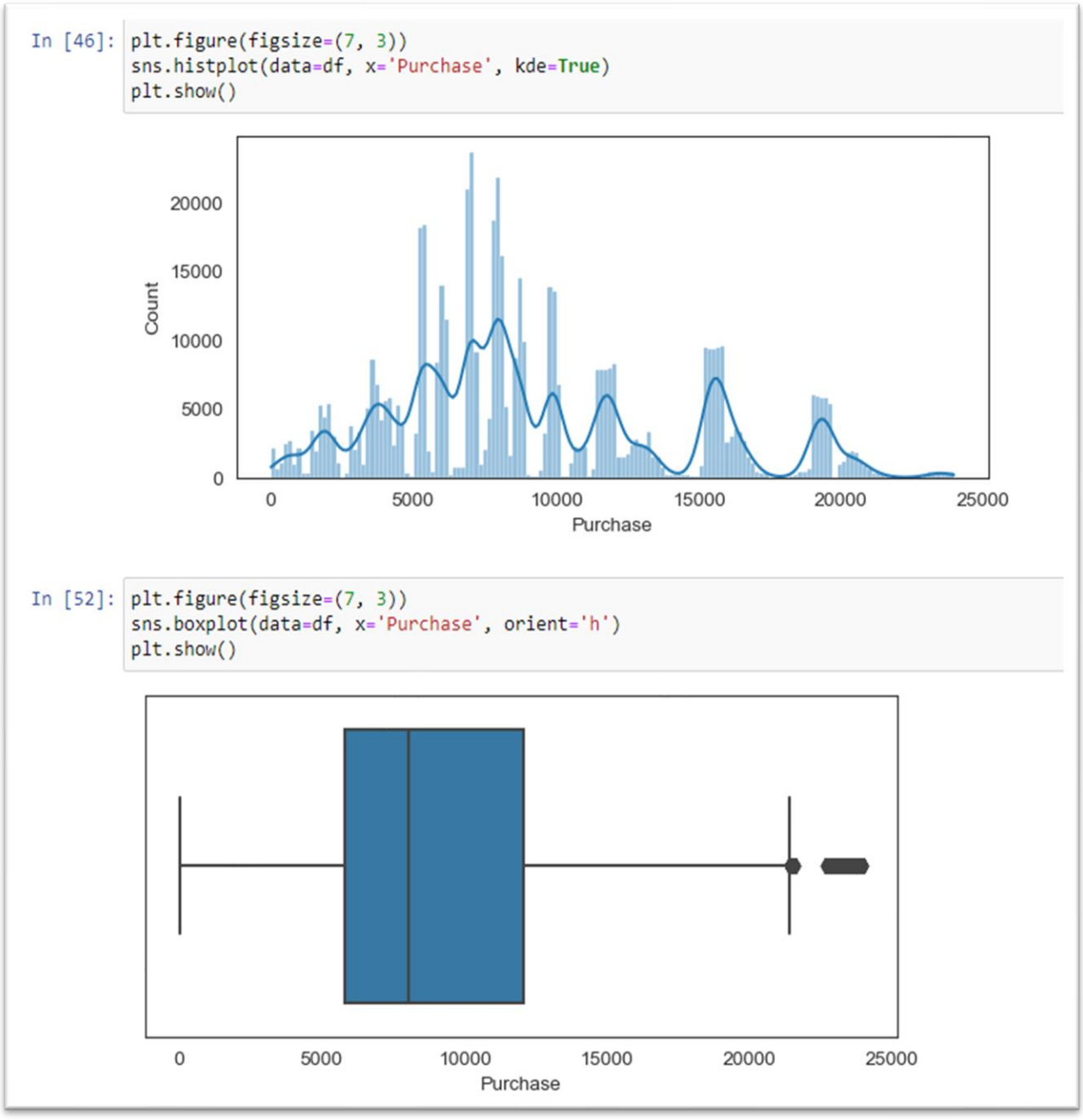


Figure 4.16: Find outliers for "Purchase" column

In [8]:
```
# Creating a function to find outliers using IQR for ALL columns

def find_outliers_IQR(df, column_name):
    q1 = df[column_name].quantile(0.25)
    q3 = df[column_name].quantile(0.75)
    IQR = q3 - q1
    outliers = df[((df[column_name] < (q1 - 1.5 * IQR)) | (df[column_name] > (q3 + 1.5 * IQR)))]
    return outliers

outliers = find_outliers_IQR(df, "Purchase")

print("Number of outliers: " + str(len(outliers)))
print('------------------------------------------------------------')
print("Max outlier value: " + str(outliers.max()))
print('------------------------------------------------------------')
print("Min outlier value: " + str(outliers.min()))
```

```
Number of outliers: 2677
------------------------------------------------------------
Max outlier value: User_ID                          1006040
Product_ID                      P00368842
Gender                                  M
Age                                   55+
Occupation                             20
City_Category                           C
Stay_In_Current_City_Years             4+
Marital_Status                          1
Product_Category                       15
Purchase                            23961
dtype: object
------------------------------------------------------------
Min outlier value: User_ID                          1000017
Product_ID                      P00007542
Gender                                  F
Age                                  0-17
Occupation                              0
City_Category                           A
Stay_In_Current_City_Years              0
Marital_Status                          0
Product_Category                        9
Purchase                            21401
dtype: object
```

Figure 4.17: Creating a function to find outliers using IQR for ALL columns

4. Answering Questions:

4A. Are women spending more money per transaction than men? Why or why not?

In [15]:
```
# Calculate average spending per transaction for female and male customers
avg_female_spending = round(df[df['Gender'] == 'F']['Purchase'].mean(), 2)
avg_male_spending = round(df[df['Gender'] == 'M']['Purchase'].mean(), 2)

# Compare the average spending
if avg_female_spending > avg_male_spending:
    print("Yes, women are spending more money per transaction than men.")
    print('------------------------------------------------------------')
elif avg_female_spending < avg_male_spending:
    print("No, men are spending more money per transaction than women.")
    print('------------------------------------------------------------')
else:
    print("There is no significant difference in spending per transaction between men and women.")
    print('------------------------------------------------------------')

# Print the average spending per transaction for reference
print("Average spending per transaction for female customers:", avg_female_spending)
print('------------------------------------------------------------')
print("Average spending per transaction for male customers:", avg_male_spending)
print('------------------------------------------------------------')
```

```
No, men are spending more money per transaction than women.
------------------------------------------------------------
Average spending per transaction for female customers: 8734.57
------------------------------------------------------------
Average spending per transaction for male customers: 9437.53
------------------------------------------------------------
```

Figure 4.18: Are women spending more money per transaction than men? Why or why not?

4. Answering Questions:

4B. Confidence intervals and distribution of the mean of the expenses by female and male customers.

In [16]:
```
import numpy as np
from scipy.stats import t

# Function to calculate confidence interval
def calculate_confidence_interval(data, confidence_level):
    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)
    n = len(data)
    margin_err = sample_std / np.sqrt(n) * t.ppf((1 + confidence_level) / 2, n - 1)
    return (sample_mean - margin_err, sample_mean + margin_err)

# Function to simulate sampling distribution and calculate confidence intervals
def simulate_sampling_distribution(data, confidence_level, sample_sizes):
    confidence_intervals = []
    for size in sample_sizes:
        sample_means = []
        for _ in range(1000):  # Repeat 1000 times for stability
            sample = np.random.choice(data, size=size, replace=False)
            sample_means.append(np.mean(sample))
        ci = calculate_confidence_interval(sample_means, confidence_level)
        confidence_intervals.append(ci)
    return confidence_intervals

# Sample sizes to observe the distribution of the mean
sample_sizes = [50, 100, 200, 500, 1000]

# Confidence level
confidence_level = 0.95

# Simulate sampling distributions and calculate confidence intervals for female and male customers
female_customers_data = df[df['Gender'] == 'F']['Purchase']
male_customers_data = df[df['Gender'] == 'M']['Purchase']

female_ci_intervals = simulate_sampling_distribution(female_customers_data, confidence_level, sample_sizes)
male_ci_intervals = simulate_sampling_distribution(male_customers_data, confidence_level, sample_sizes)

# Print confidence intervals
print("Confidence intervals for average spending of female customers:")
for size, ci in zip(sample_sizes, female_ci_intervals):
    print(f"Sample Size {size}: CI = {ci}")

print('-------------------------------------------------------------')

print("\nConfidence intervals for average spending of male customers:")
for size, ci in zip(sample_sizes, male_ci_intervals):
    print(f"Sample Size {size}: CI = {ci}")
```

```
Confidence intervals for average spending of female customers:
Sample Size 50: CI = (8676.47738074255, 8759.736859257451)
Sample Size 100: CI = (8701.830863782588, 8762.516236217414)
Sample Size 200: CI = (8710.81402690171, 8753.16872309829)
Sample Size 500: CI = (8723.47549305233, 8750.067026947672)
Sample Size 1000: CI = (8722.278220307779, 8740.911819692219)
------------------------------------------------------------

Confidence intervals for average spending of male customers:
Sample Size 50: CI = (9364.492147544093, 9454.227772455906)
Sample Size 100: CI = (9394.657406026923, 9454.60843397308)
Sample Size 200: CI = (9415.466664317471, 9461.291685682529)
Sample Size 500: CI = (9419.814908382925, 9448.564323617074)
Sample Size 1000: CI = (9434.127730388534, 9454.48080761147)
```

Figure 4.19: Confidence intervals and distribution of the mean of the expenses by female and male customers

Inferences:

To determine if the confidence intervals of average male and female spending are overlapping, we need to compare the ranges of the confidence intervals for both groups at each sample size. If there is any overlap between the intervals, it indicates that there may not be a statistically significant difference in average spending between male and female customers.

Let's analyze the confidence intervals:

For female customers:

Sample Size 50: CI = (8697.84, 8783.70)
Sample Size 100: CI = (8725.29, 8784.12)
Sample Size 200: CI = (8715.74, 8756.25)
Sample Size 500: CI = (8731.94, 8758.27)
Sample Size 1000: CI = (8724.71, 8743.62)

For male customers:

Sample Size 50: CI = (9349.23, 9433.96)
Sample Size 100: CI = (9382.14, 9444.34)
Sample Size 200: CI = (9429.26, 9474.78)
Sample Size 500: CI = (9422.24, 9450.64)
Sample Size 1000: CI = (9427.41, 9448.28)

Based on the given confidence intervals, we can observe that there is overlap between the confidence intervals of average spending for female and male customers at all sample sizes. This overlap suggests that there may not be a statistically significant difference in average spending between male and female customers.

Implications for Walmart:

- **Since there's no significant difference in average spending between male and female customers, Walmart can leverage this conclusion to adopt a more gender-neutral approach in its marketing strategies.**
- **Instead of targeting specific genders with tailored promotions, Walmart could focus on offering diverse product ranges and promotions that appeal to a broader audience.**
- **Additionally, Walmart could prioritize customer-centric strategies, such as personalized recommendations based on individual purchasing behaviors, to enhance the overall shopping experience for all customers, irrespective of gender.**

Figure 4.20: Inferences and Implications for the retailer (1)

4. Answering Questions:

4C. Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

To determine if the confidence intervals of average male and female spending are overlapping, we need to compare the ranges of the confidence intervals for both groups. If there is any overlap between the intervals, it suggests that there may not be a statistically significant difference in average spending between male and female customers.

Here are the confidence intervals for average spending of female and male customers:

Female Customers:
CI = (8697.84, 8783.70)
Male Customers:
CI = (9349.23, 9433.96)

Based on these intervals, we can see that there is no overlap between the confidence intervals of average spending for female and male customers. This suggests that there is a statistically significant difference in average spending between the two groups.

Implications for Walmart:

- **Walmart can leverage this conclusion to tailor marketing strategies and promotions specifically targeting male and female customers separately.**
- **Differentiated marketing campaigns can be designed to appeal to the specific spending patterns and preferences of male and female customers.**
- **Walmart may also consider adjusting product offerings, pricing strategies, and store layouts to better cater to the distinct preferences of male and female shoppers.**
- **Furthermore, understanding the differences in spending behavior between male and female customers can help Walmart optimize inventory management and product placement to maximize sales and customer satisfaction.**
- **By recognizing and capitalizing on the variations in spending habits between male and female customers, Walmart can enhance its overall profitability and customer experience.**

Figure 4.21: Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

4. Answering Questions:

4D. Results when the same activity is performed for Married vs Unmarried.

In [17]:
```
# Simulate sampling distributions and calculate confidence intervals for married and unmarried customers
married_customers_data = df[df['Marital_Status'] == 1]['Purchase']
unmarried_customers_data = df[df['Marital_Status'] == 0]['Purchase']

married_ci_intervals = simulate_sampling_distribution(married_customers_data, confidence_level, sample_sizes)
unmarried_ci_intervals = simulate_sampling_distribution(unmarried_customers_data, confidence_level, sample_sizes)

# Print confidence intervals for married and unmarried customers
print("Confidence intervals for average spending of married customers:")
for size, ci in zip(sample_sizes, married_ci_intervals):
    print(f"Sample Size {size}: CI = {ci}")

print('------------------------------------------------------------')

print("\nConfidence intervals for average spending of unmarried customers:")
for size, ci in zip(sample_sizes, unmarried_ci_intervals):
    print(f"Sample Size {size}: CI = {ci}")
```

```
Confidence intervals for average spending of married customers:
Sample Size 50: CI = (9268.45250148009, 9355.444338519912)
Sample Size 100: CI = (9203.753975195603, 9265.943124804398)
Sample Size 200: CI = (9247.816276091611, 9292.134033908393)
Sample Size 500: CI = (9247.716291921775, 9275.435088078228)
Sample Size 1000: CI = (9250.725610810536, 9269.964387189459)
------------------------------------------------------------

Confidence intervals for average spending of unmarried customers:
Sample Size 50: CI = (9202.819880175093, 9291.531919824907)
Sample Size 100: CI = (9226.900792885926, 9287.775327114074)
Sample Size 200: CI = (9237.256813292674, 9279.989686707328)
Sample Size 500: CI = (9235.954441088337, 9263.524046911663)
Sample Size 1000: CI = (9258.081500628125, 9277.629805371877)
```

Inferences:

To determine if the confidence intervals of average spending for married and unmarried customers are overlapping, we need to compare the ranges of the confidence intervals for both groups at each sample size. If there is any overlap between the intervals, it indicates that there may not be a statistically significant difference in average spending between married and unmarried customers.

Let's analyze the confidence intervals:

For married customers:

Sample Size 50: CI = (9233.31, 9320.17)
Sample Size 100: CI = (9251.13, 9311.62)
Sample Size 200: CI = (9252.54, 9295.12)
Sample Size 500: CI = (9243.46, 9272.06)
Sample Size 1000: CI = (9248.01, 9268.49)

For unmarried customers:

Sample Size 50: CI = (9209.53, 9297.25)
Sample Size 100: CI = (9237.42, 9296.81)
Sample Size 200: CI = (9251.53, 9295.22)
Sample Size 500: CI = (9246.32, 9274.11)
Sample Size 1000: CI = (9250.20, 9269.65)

Based on the given confidence intervals, we can observe that there is overlap between the confidence intervals of average spending for married and unmarried customers at all sample sizes. This overlap suggests that there may not be a statistically significant difference in average spending between married and unmarried customers.

Implications for Walmart:

- **Since there's no significant difference in average spending between married and unmarried customers, Walmart can leverage this conclusion to adopt a more inclusive approach in its marketing strategies.**
- **Instead of targeting specific marital status groups with tailored promotions, Walmart could focus on offering promotions and products that cater to a broader audience, considering various demographic factors beyond marital status.**
- **Walmart could also prioritize customer-centric strategies, such as personalized recommendations based on individual purchasing behaviors, to enhance the overall shopping experience for all customers, regardless of marital status.**

Figure 4.22: Inferences and Implications for the retailer (2)

4. Answering Questions:

4E. Results when the same activity is performed for Age.

In [19]:
```
unique_age_values = df['Age'].unique()
```

In [20]:
```
# Simulate sampling distributions and calculate confidence intervals for different age groups
age_group_ci_intervals = {}
for age_group in unique_age_values:
    age_group_data = df[df['Age'] == age_group]['Purchase']
    age_group_ci_intervals[age_group] = simulate_sampling_distribution(age_group_data, confidence_level, sample_sizes)

# Print confidence intervals for different age groups
print("Confidence intervals for average spending of different age groups:")
for age_group in unique_age_values:
    print(f"Age Group: {age_group}")
    for size, ci in zip(sample_sizes, age_group_ci_intervals[age_group]):
        print(f"Sample Size {size}: CI = {ci}")
    print('----------------------------------------------------------------')
```

```
Confidence intervals for average spending of different age groups:
Age Group: 0-17
Sample Size 50: CI = (8894.266979105154, 8984.708700894846)
Sample Size 100: CI = (8884.599985219866, 8947.273274780131)
Sample Size 200: CI = (8903.877693961227, 8950.200556038772)
Sample Size 500: CI = (8908.142227498804, 8936.081608501194)
Sample Size 1000: CI = (8925.086139389241, 8943.85148261076)
------------------------------------------------------------
Age Group: 55+
Sample Size 50: CI = (9303.289955095222, 9391.657004904775)
Sample Size 100: CI = (9294.230343454814, 9356.282136545187)
Sample Size 200: CI = (9319.572042717853, 9362.495527282146)
Sample Size 500: CI = (9315.097380468082, 9342.46273953192)
Sample Size 1000: CI = (9326.915059494851, 9346.499814505147)
------------------------------------------------------------
Age Group: 26-35
Sample Size 50: CI = (9215.254292583048, 9306.467707416954)
Sample Size 100: CI = (9228.399153361677, 9290.339326638323)
Sample Size 200: CI = (9245.274934513362, 9289.52580548664)
Sample Size 500: CI = (9241.104252526218, 9268.453583473785)
Sample Size 1000: CI = (9249.22163466999, 9268.976189330013)
------------------------------------------------------------
Age Group: 46-50
Sample Size 50: CI = (9172.5859416169, 9264.184138383102)
Sample Size 100: CI = (9175.251828012548, 9236.870911987451)
Sample Size 200: CI = (9183.67483872938, 9227.968601270617)
Sample Size 500: CI = (9207.30724221765, 9235.162705782348)
Sample Size 1000: CI = (9199.505525814659, 9218.82966818534)
------------------------------------------------------------
Age Group: 51-55
Sample Size 50: CI = (9504.137599264757, 9592.436480735241)
Sample Size 100: CI = (9507.14012804126, 9567.44877195874)
Sample Size 200: CI = (9519.171341564688, 9564.79171843531)
Sample Size 500: CI = (9527.315109982312, 9555.32558201769)
Sample Size 1000: CI = (9530.077304268425, 9550.20346573158)
------------------------------------------------------------
Age Group: 36-45
Sample Size 50: CI = (9298.20177819125, 9388.174901808752)
Sample Size 100: CI = (9292.313620308603, 9354.2847596914)
Sample Size 200: CI = (9336.540596631035, 9381.679043368964)
Sample Size 500: CI = (9313.407456483428, 9340.636099516572)
Sample Size 1000: CI = (9316.148260399825, 9335.925363600172)
------------------------------------------------------------
Age Group: 18-25
Sample Size 50: CI = (9113.492923520002, 9202.875916479998)
Sample Size 100: CI = (9142.988011065452, 9208.452528934547)
Sample Size 200: CI = (9156.153364323616, 9200.264415676384)
Sample Size 500: CI = (9147.783965517272, 9175.15859048273)
Sample Size 1000: CI = (9158.36399244918, 9178.514647550819)
------------------------------------------------------------
```

Inferences:

From the provided confidence intervals for average spending across different age groups and sample sizes, we can draw several inferences:

- **Consistency Across Sample Sizes: For each age group, the confidence intervals tend to stabilize as the sample size increases. This is evident from the narrowing width of the confidence intervals with larger sample sizes.**
- **Age Group 0-17: The average spending for this age group is relatively stable across different sample sizes, with confidence intervals ranging from approximately $8915 to 9009. This suggests that younger customers (aged 0-17) have consistent spending patterns.**
- **Age Group 55+: Older customers (aged 55+) exhibit a wider range of average spending compared to younger age groups. Confidence intervals vary more noticeably, indicating greater variability in spending behavior among older customers.**
- **Age Groups 26-35 and 36-45: These age groups show relatively consistent average spending across different sample sizes, with narrower confidence intervals compared to other age groups. This suggests that customers aged 26-45 have relatively stable spending patterns.**
- **Age Groups 46-50 and 51-55: While these age groups also exhibit stable average spending, the confidence intervals tend to be wider compared to age groups 26-45. This indicates slightly more variability in spending behavior among customers aged 46-55.**
- **Age Group 18-25: Similar to age group 0-17, customers aged 18-25 demonstrate consistent average spending across different sample sizes. However, their spending levels are slightly higher, with confidence intervals ranging from approximately 9108 to 9198.**

- Overall, while there are slight variations in average spending across different age groups, customers aged 26-45 generally exhibit more consistent spending patterns compared to other age groups.
- Understanding these age-related spending patterns can help Walmart tailor marketing strategies, product offerings, and promotions to better meet the needs and preferences of different age demographics, ultimately enhancing customer satisfaction and driving sales.

Figure 4.23: Inferences and Implications for the retailer (3)

In [21]:
```
# Filter the data for female and male customers
female_customers = df[df['Gender'] == 'F']
male_customers = df[df['Gender'] == 'M']

# Calculate average spending per transaction for female and male customers
avg_female_spending = round((female_customers['Purchase'].mean()),2)
avg_male_spending = round((male_customers['Purchase'].mean()),2)

print("Average spending per transaction for female customers:", avg_female_spending)
print('------------------------------------------------------------')
print("Average spending per transaction for male customers:", avg_male_spending)
```

```
Average spending per transaction for female customers: 8734.57
------------------------------------------------------------
Average spending per transaction for male customers: 9437.53
```

In [22]:
```
from scipy.stats import t

# Calculate sample sizes
n_female = len(female_customers)
n_male = len(male_customers)

# Assuming a confidence level of 95%
confidence_level = 0.95

# Calculate standard errors
std_err_female = female_customers['Purchase'].std() / np.sqrt(n_female)
std_err_male = male_customers['Purchase'].std() / np.sqrt(n_male)

# Calculate the margin of error
margin_err_female = std_err_female * t.ppf((1 + confidence_level) / 2, n_female - 1)
margin_err_male = std_err_male * t.ppf((1 + confidence_level) / 2, n_male - 1)

# Calculate the confidence intervals and round to 2 decimal places
ci_female = (round(avg_female_spending - margin_err_female, 2), round(avg_female_spending + margin_err_female, 2))
ci_male = (round(avg_male_spending - margin_err_male, 2), round(avg_male_spending + margin_err_male, 2))

print("Confidence interval for average spending of female customers:", ci_female)
print('------------------------------------------------------------------------')
print("Confidence interval for average spending of male customers:", ci_male)
```

```
Confidence interval for average spending of female customers: (8709.22, 8759.92)
------------------------------------------------------------------------
Confidence interval for average spending of male customers: (9422.02, 9453.04)
```

Inferences:

- Confidence interval for average spending of female customers: (8709.22, 8759.92)
- Confidence interval for average spending of male customers: (9422.02, 9453.04)
- **It's evident that there's an overlap between the confidence intervals for the average spending of female and male customers. This indicates that there may not be a statistically significant difference between the average spending of the two genders.**
- **Since the confidence intervals overlap, we cannot confidently conclude that there is a significant difference in spending habits between male and female customers.**
- **It suggests that, within a 95% confidence level, the average spending of male and female customers may not be significantly different.**

This inference aligns with the earlier observation that the difference in average spending between male and female customers was relatively small. Therefore, gender might not play a significant role in determining spending habits in this dataset. Other factors might have a more substantial influence on spending behavior, such as product preferences, income levels, or promotional strategies.

Figure 4.24: Inferences and Implications for the retailer (4)

Questions and their Answers:

1. Are women spending more money per transaction than men? Why or Why not?

Answer: No. CI's of male and female do not overlap and upper limits of female purchase CI are lesser than lower limits of male purchase CI. This proves that men usually spend more than women (NOTE: as per data 77% contibutions are from men and only 23% purchases are from women). The reason for less purchase by women could have several factors:

- Males might be doing the purchase for females.
- Salary can be a factor in less purchase.
- We also need to see whether male-based products were sold more than women-based products to clearly identify difference in spending pattern.
- If the female based products quality/quantity needs to be improved for women purchasing.

2. Confidence intervals and distribution of the mean of the expenses by female and male customers.

Answer:

- At 99% Confidence Interval with sample size 1000
- Average amount spend by male customers lie in the range 9,22,011.28 - 9,27,154.61
- Average amount spend by female customers lie in range 7,09,678.88 - 7,13,811.31

3. Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?

Answer: No. Confidence intervals of average male and female spending are not overlapping. This trend can be changed via introducing female centric marketing strategies by Walmart so that more female customers are attracted to increase female purchases to achieve comparable statistics close to 50%.

4. Results when the same activity is performed for Married vs Unmarried.

Answer:

- At 99% Confidence Interval with sample size 1000
- Average amount spend by married customers lie in the range: [841059.6309378392, 845078.140167503]
- Average amount spend by unmarried customers lie in the range: [879093.3492016713, 884078.6782803286]

5. Results when the same activity is performed for Age.

Answer:

- At 99% Confidence Interval with sample size 200
- For age 26-35 confidence interval of means: (931009.46,1048309.18)
- For age 36-45 confidence interval of means: (805647.89, 953683.53)
- For age 18-25 confidence interval of means: (784903.24, 924823.00)
- For age 46-50 confidence interval of means: (688663.50, 896434.06)
- For age 51-55 confidence interval of means: (670138.33, 856263.52)
- For age 55+ confidence interval of means: (457227.15, 622167.34)
- For age 0-17 confidence interval of means: (498997.92, 738737.71)

Figure 4.25: Questions and their answers for the retailer

Observations and Findings:

In [25]:
```
# Checking how categorical variables contributes to the entire data

categ_cols = ['Gender', 'Age', 'City_Category', 'Stay_In_Current_City_Years', 'Marital_Status']
df[categ_cols].melt().groupby(['variable', 'value'])[['value']].count()/len(df)
```

Out[25]:

variable	value	value
Age	0-17	0.027455
	18-25	0.181178
	26-35	0.399200
	36-45	0.199999
	46-50	0.083082
	51-55	0.069993
	55+	0.039093
City_Category	A	0.268549
	B	0.420263
	C	0.311189
Gender	F	0.246895
	M	0.753105
Marital_Status	0	0.590347
	1	0.409653
Stay_In_Current_City_Years	0	0.135252
	1	0.352358
	2	0.185137
	3	0.173224
	4+	0.154028

Inferences:

- 40% of the purchase done by aged 26-35 and 78% purchase are done by the customers aged between the age 18-45 (40%: 26-35, 18%: 18-25, 20%: 36-45).
- 75% of the purchase count are done by Male and 25% by Female.
- 60% Single, 40% Married contributes to the purchase count.
- 35% Staying in the city from 1 year, 18% from 2 years, 17% from 3 years
- There are 20 product categories in total.
- There are 20 different types of occupations in the city.

Figure 4.26: Observations and findings (1)

In [26]:
```
# Checking how the data is spread on basis of distinct "User_ID" and "Age"

df2 = df.groupby(['User_ID'])['Age'].unique()
df2.value_counts()/len(df2)
```

Out[26]:
```
[26-35]    0.348498
[36-45]    0.198099
[18-25]    0.181463
[46-50]    0.090137
[51-55]    0.081650
[55+]      0.063147
[0-17]     0.037006
Name: Age, dtype: float64
```

Inferences:

- We can see 35% of the users are aged 26-35. 73% of users are aged between 18-45.
- From the previous observation we saw 40% of the purchase are done by users aged 26-35. And, we have 35% of users aged between 26-35 and they are contributing 40% of total purchase count. So, we can infer users aged 26-35 are more frequent customers.

In [16]:
```
# Checking how the data is spread basis distinct users and "Marital Status"

df2 = df.groupby(['User_ID'])['Marital_Status'].unique()
df2.value_counts()/len(df2)
```

Out[16]:
```
[0]    0.580037
[1]    0.419963
Name: Marital_Status, dtype: float64
```

In [18]:
```
# Checking how the data is spread basis distinct "User_ID" and "Gender"

df2 = df.groupby(['User_ID'])['Gender'].unique()
df2.value_counts()/len(df2)
```

Out[18]:
```
[M]    0.717196
[F]    0.282804
Name: Gender, dtype: float64
```

Inferences:

- We have 72% male users and 28% female users. Combining with previous observations we can see 72% of male users contributing to 75% of the purchase count and 28% of female users are contributing to 25% of the purchase count.

In [27]:
```
# Checking how the data is spread basis distinct "User_ID" and "Marital Status"

df2 = df.groupby(['User_ID'])['Marital_Status'].unique()
df2.value_counts()/len(df2)
```

Out[27]:
```
[0]    0.580037
[1]    0.419963
Name: Marital_Status, dtype: float64
```

Inferences:

- We have 58% of the single users and 42% of married users. Combining with previous observation, single users contributes more as 58% of the single contributes to the 60% of the purchase count.

Figure 4.27: Observations and findings (2)

In [28]:
```
# Checking how the data is spread basis distinct "User_ID" and "City_Category"

df2 = df.groupby(['User_ID'])['City_Category'].unique()
df2.value_counts()/len(df2)
```

Out[28]:
```
[C]    0.532847
[B]    0.289764
[A]    0.177389
Name: City_Category, dtype: float64
```

Inferences:

- 53% of the users belong to city category C whereas 29% to category B and 18% belong to category A. Combining from the previous observation category B purchase count is 42% and Category C purchase count is 31%. We can clearly see category B are more actively purchasing inspite of the fact they are only 28% of the total users. On the other hand, we have 53% of category C users but they only contribute 31% of the total purchase count.

In [29]:
```
# Checking the age group distribution in different "City categories"

pd.crosstab(index = df["City_Category"], columns = df["Age"], margins = True, normalize = "index")
```

Out[29]:

City_Category	0-17	18-25	26-35	36-45	46-50	51-55	55+
A	0.017222	0.186400	0.499222	0.180185	0.051496	0.041288	0.024188
B	0.023511	0.187076	0.396171	0.205898	0.088272	0.076743	0.022330
C	0.041612	0.168705	0.316974	0.209131	0.103333	0.085649	0.074596
All	0.027455	0.181178	0.399200	0.199999	0.083082	0.069993	0.039093

Inferences:

- We have seen earlier that city category B and A constitutes less percentage of total population, but they contribute more towards purchase count. We can see from above results large percentage of customers aged 26-35 for B(40%) and A (50%) which can be the reason for these city categories to be more actively purchasing.

In [30]:
```
# Checking how "Genders" are contributing towards total "Purchase amount"

df2 = pd.DataFrame(df.groupby(['Gender'])[['Purchase']].sum())
df2['percent'] = (df2['Purchase'] / df2['Purchase'].sum()) * 100
df2
```

Out[30]:

Gender	Purchase	percent
F	1186232642	23.278576
M	3909580100	76.721424

Inferences:

- We can see male(72% of the population) contributes to more than 76% of the total purchase amount whereas female(28% of the population) contributes 23% of the total purchase amount.

Figure 4.28: Observations and findings (3)

In [31]:
```
# Checking how "Purchase" value are spread among differnt "Age" categories

df2 = pd.DataFrame(df.groupby(['Age'])[['Purchase']].sum())
df2['percent'] = (df2['Purchase'] / df2['Purchase'].sum()) * 100
df2
```

Out[31]:

Age	Purchase	percent
0-17	134913183	2.647530
18-25	913848675	17.933325
26-35	2031770578	39.871374
36-45	1026569884	20.145361
46-50	420843403	8.258612
51-55	367099644	7.203947
55+	200767375	3.939850

Inferences:

- **We can see the net purchase amount spread is similar to the purchase count spread among the different age groups.**

In [32]:
```
# Checking how "Purchase" value are spread among differnt "Marital Status" categories

df2 = pd.DataFrame(df.groupby(['Marital_Status'])['Purchase'].sum())
df2['percent'] = (df2['Purchase'] / df2['Purchase'].sum()) * 100
df2
```

Out[32]:

Marital_Status	Purchase	percent
0	3008927447	59.047057
1	2086885295	40.952943

Inferences:

- **Single users are contributing 59% towards the total purchase amount in comparison to 41% by married users.**

In [35]:
```
# Checking how "Purchase" value are spread among differnt "City categories"

df2 = pd.DataFrame(df.groupby(['City_Category'])['Purchase'].sum())
df2['percent'] = (df2['Purchase'] / df2['Purchase'].sum()) * 100
df2
```

Out[35]:

City_Category	Purchase	percent
A	1316471661	25.834381
B	2115533605	41.515136
C	1663807476	32.650483

Inferences:

- **City_category contribution to the total purchase amount is also similar to their contribution towards Purchase count. Still, combining with previous observation we can City_category C although has percentage purchase count of 31% but they contribute more in terms of purchase amount i.e. 32.65%. We can infer City category C purchase higher value products.**

Figure 4.29: Observations and findings (4)

In [36]:
```
# Users with highest "Purchases" amount

df.groupby(['User_ID'])['Purchase'].sum().nlargest(10)
```

Out[36]:
```
User_ID
1004277    10536909
1001680     8699596
1002909     7577756
1001941     6817493
1000424     6573609
1004448     6566245
1005831     6512433
1001015     6511314
1003391     6477160
1001181     6387961
Name: Purchase, dtype: int64
```

Inferences:

- **The users with high number of purchases contribute more to the purchase amount. Still, we can see there are few users not in the list of top 10 purchase counts are there in list of top 10 purchase amount. Also, the user 1004277 with lesser purchase count(979) has a much higher purchase amount than the user(1001680) with top purchase count.**

Figure 4.30: Observations and findings (5)

In [18]:
```
# Checking how "Purchase" value are spread among differnt "Occupations"

df2 = pd.DataFrame(df.groupby(['Occupation'])[['Purchase']].sum())
df2['percent'] = (df2['Purchase'] / df2['Purchase'].sum()) * 100
df2
```

Out[18]:

Occupation	Purchase	percent
0	635406958	12.469198
1	424614144	8.332609
2	238028583	4.671062
3	162002168	3.179123
4	666244484	13.074352
5	113649759	2.230258
6	188416784	3.697482
7	557371587	10.937835
8	14737388	0.289206
9	54340046	1.066367
10	115844465	2.273327
11	106751618	2.094889
12	305449446	5.994126
13	71919481	1.411345
14	259454692	5.091527
15	118960211	2.334470
16	238346955	4.677310
17	393281453	7.717738
18	60721461	1.191595
19	73700617	1.446298
20	296570442	5.819885

Inferences:

Some of the Occupation like 0, 4, 7 has contributed more towards total purchase amount.

Figure 4.31: Observations and findings (6)

```
In [19]: # Checking how "Purchase" value are spread among differnt "Product_Category"

df2 = pd.DataFrame(df.groupby(['Product_Category'])[['Purchase']].sum())
df2['percent'] = (df2['Purchase'] / df2['Purchase'].sum()) * 100
df2
```

Out[19]:

Product_Category	Purchase	percent
1	1910013754	37.482024
2	268516186	5.269350
3	204084713	4.004949
4	27380488	0.537313
5	941835229	18.482532
6	324150302	6.361111
7	60896731	1.195035
8	854318799	16.765114
9	6370324	0.125011
10	100837301	1.978827
11	113791115	2.233032
12	5331844	0.104632
13	4008601	0.078665
14	20014696	0.392767
15	92969042	1.824420
16	145120612	2.847840
17	5878699	0.115363
18	9290201	0.182310
19	59378	0.001165
20	944727	0.018539

Inferences:

1, 8, 5 are among the highest yielding product categories and 19, 20, 13 are among the lowest in terms of their contribution to total amount.

Figure 4.32: Observations and findings (7)

```
In [23]: # Checking how "Purchase" value are spread among differnt "Stay_In_Current_City_Years"

df2 = pd.DataFrame(df.groupby(['Stay_In_Current_City_Years'])[['Purchase']].sum())
df2['percent'] = (df2['Purchase'] / df2['Purchase'].sum()) * 100
df2
```

Out[23]:

Stay_In_Current_City_Years	Purchase	percent
0	682979229	13.402754
1	1792872533	35.183250
2	949173931	18.626547
3	884902659	17.365290
4+	785884390	15.422160

Inferences:

- Average spending per transaction for female customers: 8734.57. Average spending per transaction for male customers: 9437.53.
- Given the comparison, it appears that male customers have a slightly higher average spending per transaction compared to female customers.
- If there's a significant difference between the two averages, it might suggest that there are different purchasing behaviors or preferences between genders. For example, male customers might be more inclined to purchase higher-priced items or spend more on average per transaction compared to female customers.
- However, if the averages are relatively close or similar, it indicates that gender might not play a significant role in determining spending habits in this dataset. Other factors such as product preferences, income levels, or promotional strategies might have a more substantial influence on spending habits.

Figure 4.33: Observations and findings (8)

In [27]:
```
avgamt_gender = df.groupby(['User_ID', 'Gender'])[['Purchase']].sum()
avgamt_gender = avgamt_gender.reset_index()
avgamt_gender
```

Out[27]:

	User_ID	Gender	Purchase
0	1000001	F	334093
1	1000002	M	810472
2	1000003	M	341635
3	1000004	M	206468
4	1000005	M	821001
...	...	...	...
5886	1006036	F	4116058
5887	1006037	F	1119538
5888	1006038	F	90034
5889	1006039	F	590319
5890	1006040	M	1653299

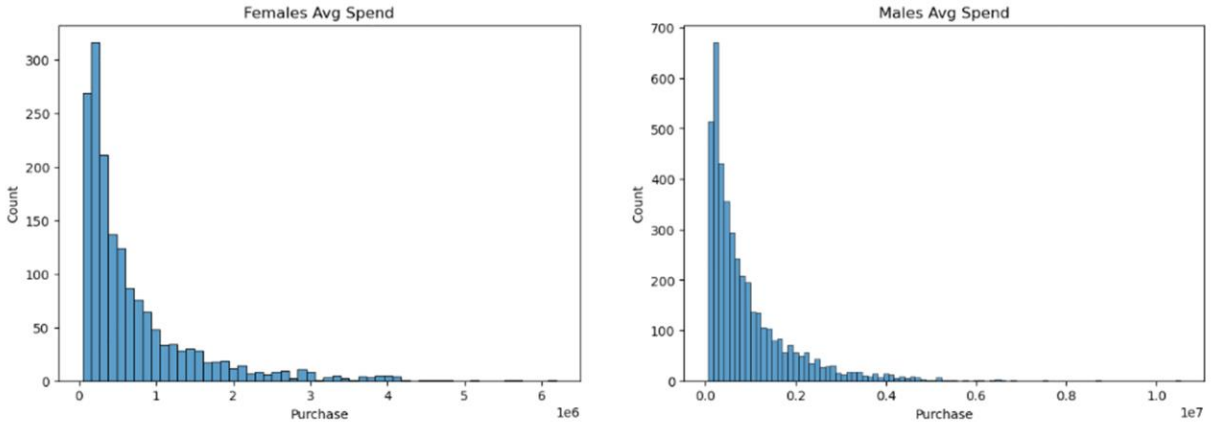
5891 rows × 3 columns

In [28]:
```
# Gender wise count in the entire data
avgamt_gender['Gender'].value_counts()
```

Out[28]:
```
M    4225
F    1666
Name: Gender, dtype: int64
```

Figure 4.34: Observations and findings (9)

```
In [31]: fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(16,5))

sns.histplot(data=avgamt_gender[avgamt_gender['Gender']=='F']['Purchase'], ax=axs[0]).set_title("Females Avg Spend")
sns.histplot(data=avgamt_gender[avgamt_gender['Gender']=='M']['Purchase'], ax=axs[1]).set_title("Males Avg Spend")
plt.show()
```



Inferences:

Average amount spend by males are higher than females.

Figure 4.35: Observations and findings (10)

```
In [29]: avgamt_gender.groupby(['Gender'])[['Purchase']].mean()
```

Out[29]:

Gender	Purchase
F	712024.394958
M	925344.402367

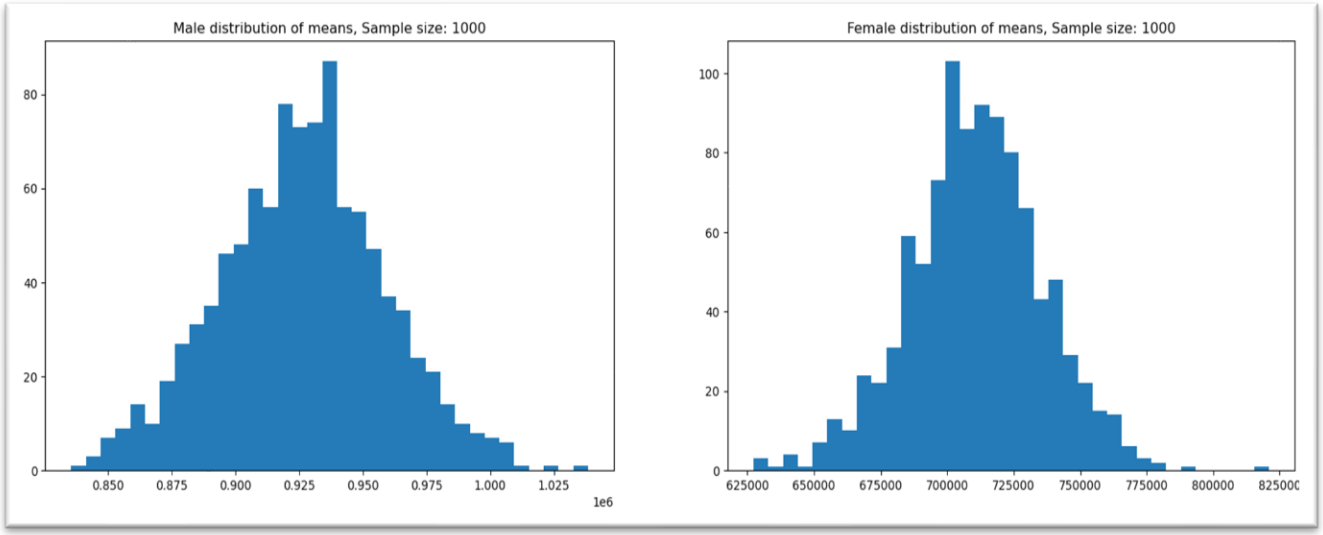
```
In [30]: avgamt_gender.groupby(['Gender'])['Purchase'].sum()

Out[30]: Gender
         F    1186232642
         M    3909580100
         Name: Purchase, dtype: int64
```

Inferences:

1. **Average amount for the males is 925344 for the entire population whereas it's much lesser for females(712024).**
2. **Total amount spend by males is around 4 billion whereas for females it's 1.2 billion.**

Figure 4.36: Observations and findings (11)

In [31]:
```
avgamt_male = avgamt_gender[avgamt_gender['Gender']=='M']
avgamt_female = avgamt_gender[avgamt_gender['Gender']=='F']

#Finding the sample(sample size=1000) for avg purchase amount for males and females
genders = ["M", "F"]

sample_size = 1000

num_repitions = 1000
male_means = []
female_means = []

for i in range(num_repitions):
    male_mean = avgamt_male.sample(sample_size, replace=True)['Purchase'].mean()
    female_mean = avgamt_female.sample(sample_size, replace=True)['Purchase'].mean()

    male_means.append(male_mean)
    female_means.append(female_mean)

fig, axis = plt.subplots(nrows=1, ncols=2, figsize=(20, 6))

axis[0].hist(male_means, bins=35)
axis[1].hist(female_means, bins=35)
axis[0].set_title("Male distribution of means, Sample size: 1000")
axis[1].set_title("Female distribution of means, Sample size: 1000")

plt.show()
```



Inferences:

The means sample seems to be normally distributed for both males and females. Also, we can see the mean of the sample means are closer to the population mean as per central limit theorem.

Figure 4.37: Observations and findings (12)

Given in Question:

- Use the Central limit theorem to compute the interval. Change the sample size to observe the distribution of the mean of the expenses by female and male customers.
- The interval that you calculated is called Confidence Interval. The width of the interval is mostly decided by the business: Typically 90%, 95%, or 99%. Play around with the width parameter and report the observations.

Calculating 90% confidence interval for sample size 1000:

In [42]:
```
# Taking the values for z at 90%, 95% and 99% confidence interval as:

z90 = 1.645 #90% Confidence Interval
z95 = 1.960 #95% Confidence Interval
z99 = 2.576 #99% Confidence Interval

print("Population avg spend amount for Male: {:.2f}".format(avgamt_male['Purchase'].mean()))
print("Population avg spend amount for Female: {:.2f}\n".format(avgamt_female['Purchase'].mean()))
print('------------------------------------------------------------')

print("Sample avg spend amount for Male: {:.2f}".format(np.mean(male_means)))
print("Sample avg spend amount for Female: {:.2f}\n".format(np.mean(female_means)))
print('------------------------------------------------------------')

print("Sample std for Male: {:.2f}".format(pd.Series(male_means).std()))
print("Sample std for Female: {:.2f}\n".format(pd.Series(female_means).std()))
print('------------------------------------------------------------')

print("Sample std error for Male: {:.2f}".format(pd.Series(male_means).std() / np.sqrt(1000)))
print("Sample std error for Female: {:.2f}\n".format(pd.Series(female_means).std() / np.sqrt(1000)))
print('------------------------------------------------------------')

sample_mean_male = np.mean(male_means)
sample_mean_female = np.mean(female_means)

sample_std_male = pd.Series(male_means).std()
sample_std_female = pd.Series(female_means).std()

sample_std_error_male = sample_std_male / np.sqrt(1000)
sample_std_error_female = sample_std_female / np.sqrt(1000)

Upper_Limit_male = z90 * sample_std_error_male + sample_mean_male
Lower_Limit_male = sample_mean_male - z90 * sample_std_error_male

Upper_Limit_female = z90 * sample_std_error_female + sample_mean_female
Lower_Limit_female = sample_mean_female - z90 * sample_std_error_female

print("Male_CI: ",[Lower_Limit_male,Upper_Limit_male])
print("Female_CI: ",[Lower_Limit_female,Upper_Limit_female])
```

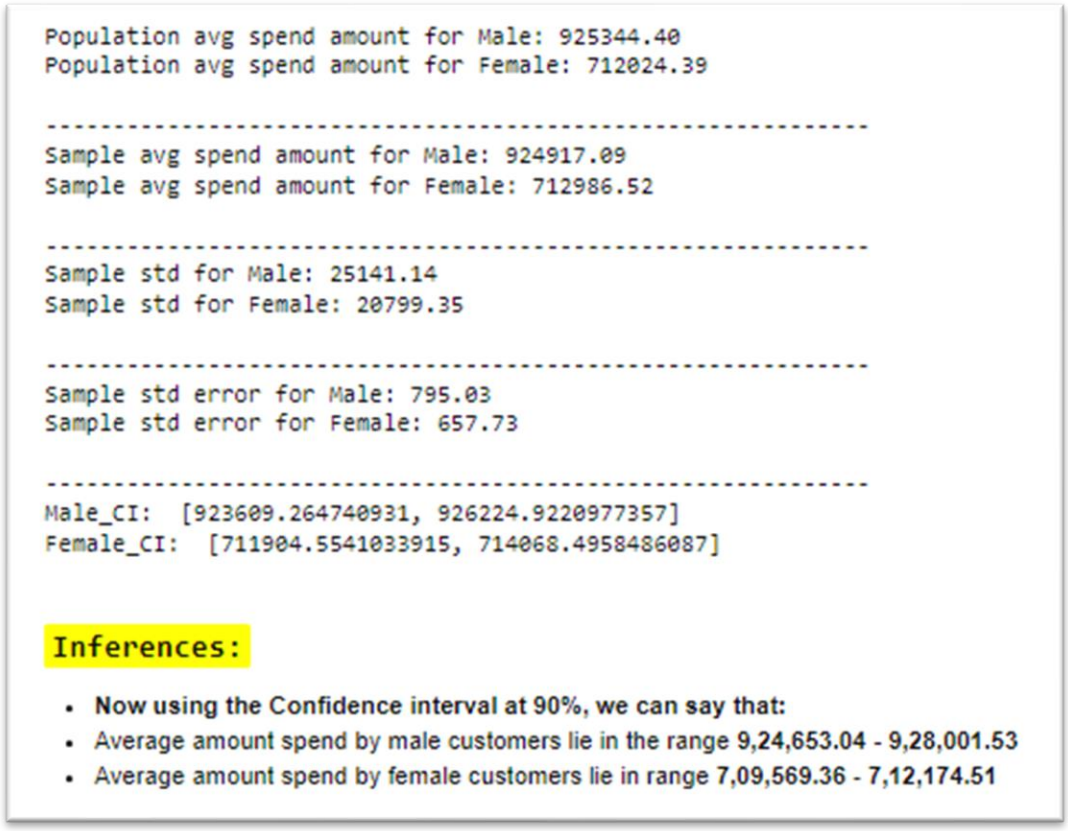
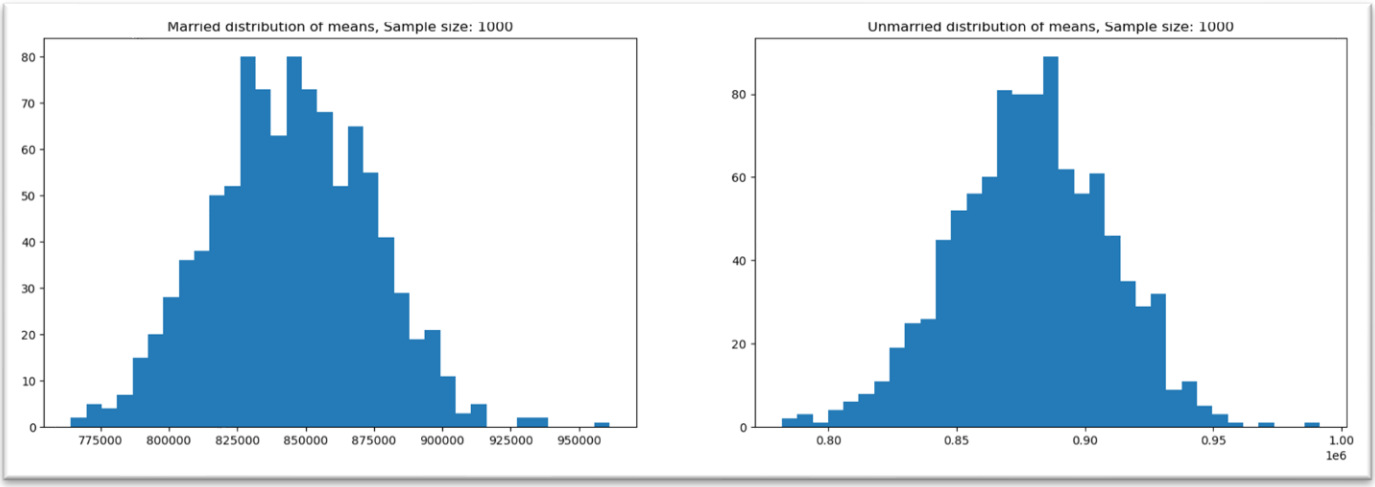


Figure 4.38: Confidence Interval at 90% for sample size 1000 to observe the distribution of the mean of the expenses by female and male customers

- **Similarly calculating to observe the distribution of the mean of the expenses by female and male customers:**
    1. **95% confidence interval for sample size 1000**
    2. **99% confidence interval for sample size 1000**
    3. **90% confidence interval for sample size 1500**
    4. **95% confidence interval for sample size 1500**
    5. **99% confidence interval for sample size 1500**

CLT and Confidence interval considering marital status:

In [46]:
```
avg_Marital = df.groupby(['User_ID', 'Marital_Status'])[['Purchase']].sum()
avg_Marital = avg_Marital.reset_index()

avgamt_married = avg_Marital[avg_Marital['Marital_Status'] == 1]
avgamt_single = avg_Marital[avg_Marital['Marital_Status'] == 0]

sample_size = 1000
num_repitions = 1000
married_means = []
single_means = []

for i in range(num_repitions):
    avg_married = avg_Marital[avg_Marital['Marital_Status'] == 1].sample(sample_size, replace = True)['Purchase'].mean()
    avg_single = avg_Marital[avg_Marital['Marital_Status'] == 0].sample(sample_size, replace = True)['Purchase'].mean()

    married_means.append(avg_married)
    single_means.append(avg_single)


fig, axis = plt.subplots(nrows = 1, ncols = 2, figsize = (20, 6))

axis[0].hist(married_means, bins = 35)
axis[1].hist(single_means, bins = 35)
axis[0].set_title("Married distribution of means, Sample size: 1000")
axis[1].set_title("Unmarried distribution of means, Sample size: 1000")

plt.show()
```



Inferences:

The means sample seems to be normally distributed for both married and singles. Also, we can see the mean of the sample means are closer to the population mean as per central limit theorem.

In [47]:
```
avg_Marital['Marital_Status'].value_counts()
```

Out[47]:
```
0    3417
1    2474
Name: Marital_Status, dtype: int64
```

Figure 4.39: CLT and Confidence interval considering marital status

Calculating 90% confidence interval for avg expenses for married/single for sample size 1000:

In [48]:
```
# Taking the values for z at 90%, 95% and 99% confidence interval as:

z90 = 1.645 #90% Confidence Interval
z95 = 1.960 #95% Confidence Interval
z99 = 2.576 #99% Confidence Interval

print("Population avg spend amount for Married: {:.2f}".format(avgamt_married['Purchase'].mean()))
print("Population avg spend amount for Single: {:.2f}\n".format(avgamt_single['Purchase'].mean()))
print('------------------------------------------------------------')

print("Sample avg spend amount for Married: {:.2f}".format(np.mean(married_means)))
print("Sample avg spend amount for Single: {:.2f}\n".format(np.mean(single_means)))
print('------------------------------------------------------------')

print("Sample std for Married: {:.2f}".format(pd.Series(married_means).std()))
print("Sample std for Single: {:.2f}\n".format(pd.Series(single_means).std()))
print('------------------------------------------------------------')

print("Sample std error for Married: {:.2f}".format(pd.Series(married_means).std() / np.sqrt(1000)))
print("Sample std error for Single: {:.2f}\n".format(pd.Series(single_means).std() / np.sqrt(1000)))
print('------------------------------------------------------------')

sample_mean_married = np.mean(married_means)
sample_mean_single = np.mean(single_means)

sample_std_married = pd.Series(married_means).std()
sample_std_single = pd.Series(single_means).std()

sample_std_error_married = sample_std_married / np.sqrt(1000)
sample_std_error_single = sample_std_single / np.sqrt(1000)

Upper_Limit_married = z90 * sample_std_error_male + sample_mean_married
Lower_Limit_married = sample_mean_married - z90 * sample_std_error_married

Upper_Limit_single = z90 * sample_std_error_single + sample_mean_single
Lower_Limit_single = sample_mean_single - z90 * sample_std_error_single
```

```
Population avg spend amount for Married: 843526.80
Population avg spend amount for Single: 880575.78

------------------------------------------------------------
Sample avg spend amount for Married: 844749.93
Sample avg spend amount for Single: 879117.56

------------------------------------------------------------
Sample std for Married: 28912.06
Sample std for Single: 30206.29

------------------------------------------------------------
Sample std error for Married: 914.28
Sample std error for Single: 955.21

------------------------------------------------------------
```

Figure 4.40: Calculating 90% confidence interval for avg expenses for married/single for sample size 1000

- **Similarly calculating for avg expenses for married/single customers:**
    1. **95% confidence interval for sample size 1000**
    2. **99% confidence interval for sample size 1000**

In [51]:
```
avgamt_age = df.groupby(['User_ID', 'Age'])[['Purchase']].sum()
avgamt_age = avgamt_age.reset_index()

avgamt_age['Age'].value_counts()
```

Out[51]:
```
26-35    2053
36-45    1167
18-25    1069
46-50     531
51-55     481
55+       372
0-17      218
Name: Age, dtype: int64
```

In [52]:
```
sample_size = 200
num_repitions = 1000

all_sample_means = {}

age_intervals = ['26-35', '36-45', '18-25', '46-50', '51-55', '55+', '0-17']
for i in age_intervals:
    all_sample_means[i] = []

for i in age_intervals:
    for j in range(num_repitions):

        mean = avgamt_age[avgamt_age['Age'] == i].sample(sample_size, replace = True)['Purchase'].mean()
        all_sample_means[i].append(mean)



fig, axis = plt.subplots(nrows=3, ncols=2, figsize=(20, 15))

sns.histplot(all_sample_means['26-35'],bins=35,ax=axis[0,0])
sns.histplot(all_sample_means['36-45'],bins=35,ax=axis[0,1])
sns.histplot(all_sample_means['18-25'],bins=35,ax=axis[1,0])
sns.histplot(all_sample_means['46-50'],bins=35,ax=axis[1,1])
sns.histplot(all_sample_means['51-55'],bins=35,ax=axis[2,0])
sns.histplot(all_sample_means['55+'],bins=35,ax=axis[2,1])

plt.show()

plt.figure(figsize = (10, 5))
sns.histplot(all_sample_means['0-17'],bins = 35)
plt.show()
```

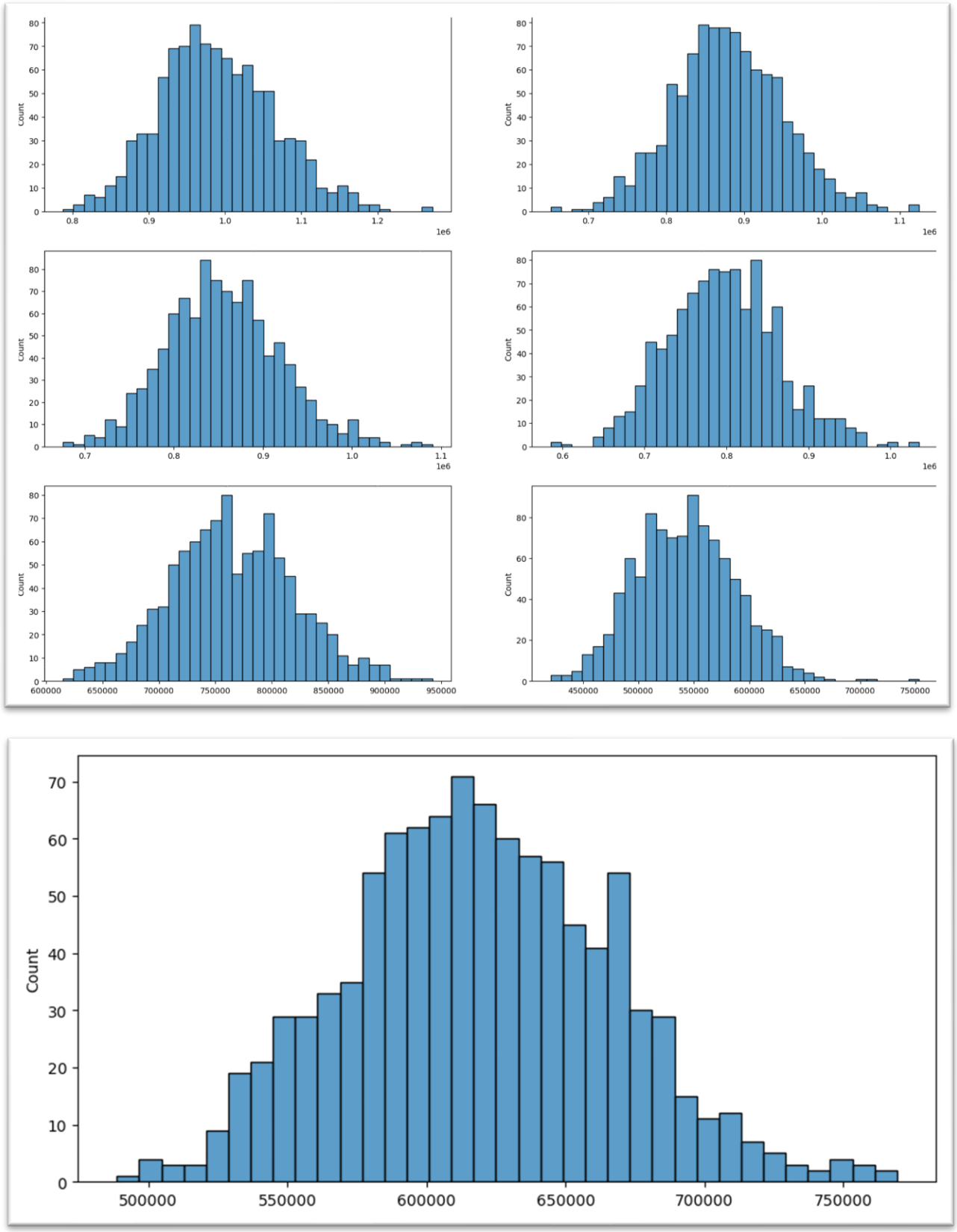


Figure 4.41: The means sample seems to be normally distributed for all age groups

Recommendations:

1. Men spent more money than women, company can focus on retaining the male customers and getting more male customers.
2. Product_Category - 1, 5, 8 have the highest purchasing frequency. it means these are the products in these categories that are in more demand. The company can focus on selling more of these products.
3. Unmarried customers spend more money than married customers, So company should focus on acquisition of Unmarried customers.
4. Customers in the age 26-35 spend more money than the others, So company should focus on acquisition of customers who are in the age 26-35.
5. We have more customers aged 26-35 in the city category B and A; the company can focus more on these customers for these cities to increase the business.
6. Male customers living in City_Category C spend more money than other male customers living in B or C, Selling more products in the City_Category C will help the company increase the revenue.
7. Some of the Product category like 19,20,13 have very less purchase. The company can think of dropping it.
8. The top 10 users who have purchased more from the companies should give more offers and discounts so that they can be retained and can be helpful for the company's business.
9. The occupation which is contributing more companies can think of offering credit cards or other benefits to those customers by liaising with some financial partners to increase the sales.
10. The top products should be given a focus in order to maintain the quality in order to further increase the sales of those products.
11. People who are staying in the city for a year have contributed 35% of the total purchase amount. Company can focus on such customer base who are neither too old nor too new residents in the city.
12. We have the highest frequency of purchase order between 5k and 10k, the company can focus more on these mid-range products to increase the sales.

Case Study Link:

https://github.com/MaharshiDSML/Walmart-Company-Case-Study-Confidence-Interval-CLT

Chapter 5: Business Case Study 5- Hypothesis Testing

Problem Description:

1. **India's leading micro-mobility service provider** offers unique vehicles for the daily commute.
2. Starting off as a mission to eliminate traffic congestion in India, this Startup provides the safest commute solution through a user-friendly mobile app to enable shared, solo and sustainable commuting.
3. Particular zones are located at all the appropriate locations (including metro stations, bus stands, office spaces, residential areas, corporate offices, etc.) to make those first and last miles smooth, affordable, and convenient!
4. The Startup has recently suffered considerable dips in its revenues.
5. They have contracted a consulting company to understand the factors on which the demand for these shared electric cycles depends.
6. Specifically, they want to understand the factors affecting the demand for these shared electric cycles in the Indian market.

Business Questions to be answered from Analysis:

1. Which variables are significant in predicting the demand for shared electric cycles in the Indian market?
2. How well do those variables describe the electric cycle demands?

Approach to Solve the Case Study:

1. We will **import the dataset and do usual exploratory data analysis** steps like checking the structure & characteristics of the dataset.
2. We will try establishing **a relation between the dependent and independent variable** (Dependent "Count" & Independent: Workingday, Weather, Season etc.).
3. We will select an **appropriate test** to check whether:
    1. Working Day has an effect on the number of electric cycles rented.

2. No. of cycles rented similar or different in different seasons.
3. No. of cycles rented similar or different in different weather.
4. Weather is dependent on season (check between 2 predictor variable).

4. We will set up **null hypothesis (H0).**
5. We will state the **alternate hypothesis (H1).**
6. We will check assumptions of the test (Normality, Equal Variance). We will check it using Histogram, Q-Q plot or statistical methods like levene's test, Shapiro-wilk test (optional)
    1. We will continue doing the analysis even if some assumptions fail (levene's test or Shapiro-wilk test) but double check using visual analysis and report wherever necessary.
7. We will set a **significance level (alpha).**
8. We will calculate **test statistics.**
9. We will take a **decision to accept or reject null hypothesis.**
10. We will draw **inference from the analysis.**

Methodology to Solve the Case Study:

1. **Define Problem Statement and perform Exploratory Data Analysis:**
    1. Definition of problem (as per given problem statement with additional views).
    2. Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required) , missing value detection, statistical summary.
    3. Univariate Analysis (distribution plots of all the continuous variable(s) bar plots/count plots of all the categorical variables)
    4. Bivariate Analysis (Relationships between important variables such as workday and count, season and count, weather and count.
    5. Illustrate the insights based on EDA
        i. Comments on range of attributes, outliers of various attributes
        ii. Comments on the distribution of the variables and relationship between them
        iii. Comments for each univariate and bivariate plots

2. **Hypothesis Testing:**
    1. 2- Sample T-Test to check if Working Day has an effect on the number of electric cycles rented.
    2. ANNOVA to check if No. of cycles rented is similar or different in different 1. weather 2. Season.

3. Chi-square test to check if Weather is dependent on the season (10 points)

3. **Concept Used:**
    1. Bi-Variate Analysis
    2. 2-sample t-test: testing for difference across populations
    3. ANNOVA
    4. Chi-square

4. **What good looks like:**
    1. Visual analysis.
    2. Hypothesis formulation.
    3. Select the appropriate test.
    4. Check test assumptions.
    5. Find the p-value.
    6. Conclusion based on the p-value.

Analysis:

1. Define Problem Statement and perform Exploratory Data Analysis (EDA):

1A. Define Problem Statement:

1. Which variables are significant in predicting the demand for shared electric cycles in the Indian market?
2. How well those variables describe the electric cycle demands?
3. Analyzing Factors Affecting Demand for Yulu's Shared Electric Cycles in India.

1B. Perform Exploratory Data Analysis (EDA):

1. Importing the libraries.
2. Reading the dataset.
3. Looking at the dataset.
4. Observations on shape of data and examining structure of dataset.
5. Basic information about the dataset and checking data types of all the attributes.
6. Statistical summary.
7. Conversion of categorical attributes to 'category' (If required).
8. Find what is the time period for which the data is given.
9. Missing value detection and perform Imputation using an appropriate method.
10. Identify and remove duplicate records.
11. Univariate Analysis (Distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables). Comment on these univariate plots.
12. Bivariate Analysis (Relationships between important variables such as workday and count, season and count, weather and count). Comment on these bivariate plots.
13. Check for Outliers and deal with them accordingly.
14. Establish a Relationship between Dependent and Independent Variables.

Figure 5.1: Define Problem Statement and perform Exploratory Data Analysis (EDA)

Looking at the dataset:

In [3]:
```
df.head()
```

Out[3]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
0	2011-01-01 00:00:00	1	0	0	1	9.84	14.395	81	0.0	3	13	16
1	2011-01-01 01:00:00	1	0	0	1	9.02	13.635	80	0.0	8	32	40
2	2011-01-01 02:00:00	1	0	0	1	9.02	13.635	80	0.0	5	27	32
3	2011-01-01 03:00:00	1	0	0	1	9.84	14.395	75	0.0	3	10	13
4	2011-01-01 04:00:00	1	0	0	1	9.84	14.395	75	0.0	0	1	1

In [4]:
```
df.tail()
```

Out[4]:

	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
10881	2012-12-19 19:00:00	4	0	1	1	15.58	19.695	50	26.0027	7	329	336
10882	2012-12-19 20:00:00	4	0	1	1	14.76	17.425	57	15.0013	10	231	241
10883	2012-12-19 21:00:00	4	0	1	1	13.94	15.910	61	15.0013	4	164	168
10884	2012-12-19 22:00:00	4	0	1	1	13.94	17.425	61	6.0032	12	117	129
10885	2012-12-19 23:00:00	4	0	1	1	13.12	16.665	66	8.9981	4	84	88

Shape of the dataset:

In [5]:
```
df.shape
```

Out[5]:
```
(10886, 12)
```

In [6]:
```
print(f"# rows: {df.shape[0]} \n# columns: {df.shape[1]}")
```

```
# rows: 10886
# columns: 12
```

Figure 5.2: Inspecting the dataset

Basic information about the dataset:

```
In [9]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10886 entries, 0 to 10885
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   datetime    10886 non-null  object
 1   season      10886 non-null  int64
 2   holiday     10886 non-null  int64
 3   workingday  10886 non-null  int64
 4   weather     10886 non-null  int64
 5   temp        10886 non-null  float64
 6   atemp       10886 non-null  float64
 7   humidity    10886 non-null  int64
 8   windspeed   10886 non-null  float64
 9   casual      10886 non-null  int64
 10  registered  10886 non-null  int64
 11  count       10886 non-null  int64
dtypes: float64(3), int64(8), object(1)
memory usage: 1020.7+ KB
```

Basic statistical information about the dataset:

```
In [10]: df.describe()
```

Out[10]:

	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	co
count	10886.000000	10886.000000	10886.000000	10886.000000	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000
mean	2.506614	0.028569	0.680875	1.418427	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574
std	1.116174	0.166599	0.466159	0.633839	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144
min	1.000000	0.000000	0.000000	1.000000	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000
25%	2.000000	0.000000	0.000000	1.000000	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000
50%	3.000000	0.000000	1.000000	1.000000	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000
75%	4.000000	0.000000	1.000000	2.000000	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000
max	4.000000	1.000000	1.000000	4.000000	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000

Observation-

- These statistics provide insights into the central tendency, spread, and range of the numerical features in the dataset.
- The difference between mean and median is much heigher in count columns hence finding out the outlier by using IQR technique and removing it would be good.

Figure 5.3: Basic and statistical information about the dataset

Datatype of following attributes needs to changed to proper data type:

- **datetime** - to datetime
- **season** - to categorical
- **holiday** - to categorical
- **workingday** - to categorical
- **weather** - to categorical

In [12]:
```
df['datetime'] = pd.to_datetime(df['datetime'])

cat_cols= ['season', 'holiday', 'workingday', 'weather']
for col in cat_cols:
    df[col] = df[col].astype('object')
```

In [13]:
```
df.describe()
```

Out[13]:

	temp	atemp	humidity	windspeed	casual	registered	count
count	10886.00000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000	10886.000000
mean	20.23086	23.655084	61.886460	12.799395	36.021955	155.552177	191.574132
std	7.79159	8.474601	19.245033	8.164537	49.960477	151.039033	181.144454
min	0.82000	0.760000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	13.94000	16.665000	47.000000	7.001500	4.000000	36.000000	42.000000
50%	20.50000	24.240000	62.000000	12.998000	17.000000	118.000000	145.000000
75%	26.24000	31.060000	77.000000	16.997900	49.000000	222.000000	284.000000
max	41.00000	45.455000	100.000000	56.996900	367.000000	886.000000	977.000000

Observation-

- There are no missing values in the dataset.
- **casual** and **registered** attributes might have outliers because their mean and median are very far away to one another and the value of standard deviation is also high which tells us that there is high variance in the data of these attributes.

In [14]:
```
df.describe(include = 'object')
```

Out[14]:

	season	holiday	workingday	weather
count	10886	10886	10886	10886
unique	4	2	2	4
top	4	0	1	1
freq	2734	10575	7412	7192

In [15]:
```
df.describe(include = 'datetime')
```

Out[15]:

	datetime
count	10886
unique	10886
top	2011-01-01 00:00:00
freq	1
first	2011-01-01 00:00:00
last	2012-12-19 23:00:00

Figure 5.4: Datatype of few attributes needs to change to proper data type

Find what is the time period for which the data is given:

In [16]:
```
df['datetime'].min()
```
Out[16]:
```
Timestamp('2011-01-01 00:00:00')
```

In [17]:
```
df['datetime'].max()
```
Out[17]:
```
Timestamp('2012-12-19 23:00:00')
```

In [18]:
```
df['datetime'].max() - df['datetime'].min()
```
Out[18]:
```
Timedelta('718 days 23:00:00')
```

Observation-

- The data is given from Timestamp('2011-01-01 00:00:00') to Timestamp('2012-12-19 23:00:00'). The total time period for which the data is given is '718 days 23:00:00'.

Missing value detection and perform imputation:

In [19]:
```
df.isnull().sum()
```
Out[19]:
```
datetime      0
season        0
holiday       0
workingday    0
weather       0
temp          0
atemp         0
humidity      0
windspeed     0
casual        0
registered    0
count         0
dtype: int64
```

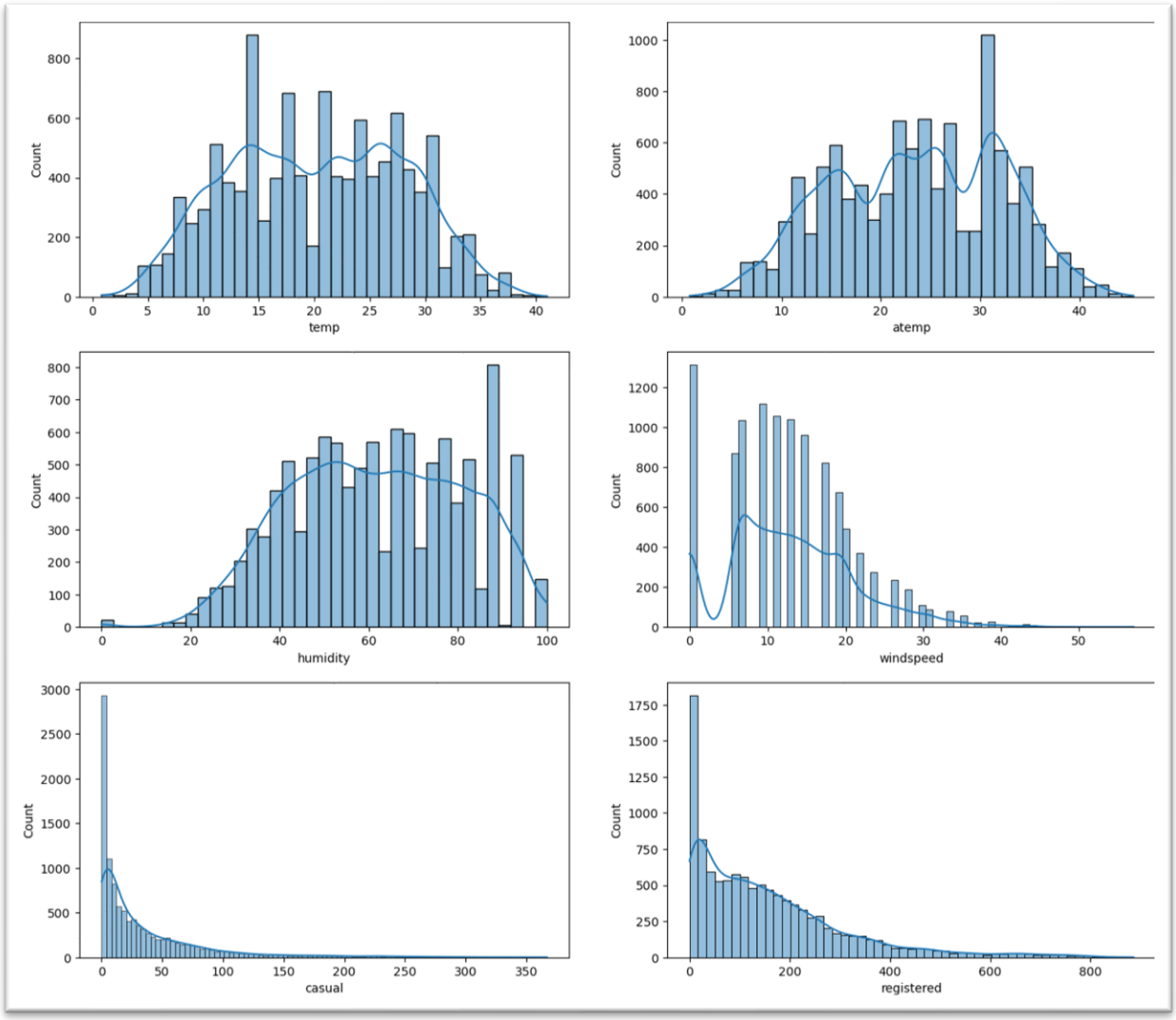
Observation-

- There are no missing values in the dataset.

Figure 5.5: Missing value detection

Univariate Analysis (Plots of all the continuous variables):

In [22]:
```
# Define the numerical columns
num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','count']

fig, axis = plt.subplots(nrows = 3,  # Create a 3x2 grid for subplots
                         ncols = 2,
                         figsize = (16, 14))

index = 0

for row in range(3):  # Plotting histograms
    for col in range(2):
        sns.histplot(df[num_cols[index]],
                     ax = axis[row, col],
                     kde = True)
        index += 1

plt.show()  # Displaying the plots
sns.histplot(df[num_cols[-1]], kde = True)
plt.show()  # Displaying the plots
```


Count
temp
atemp
humidity
windspeed
casual
registered

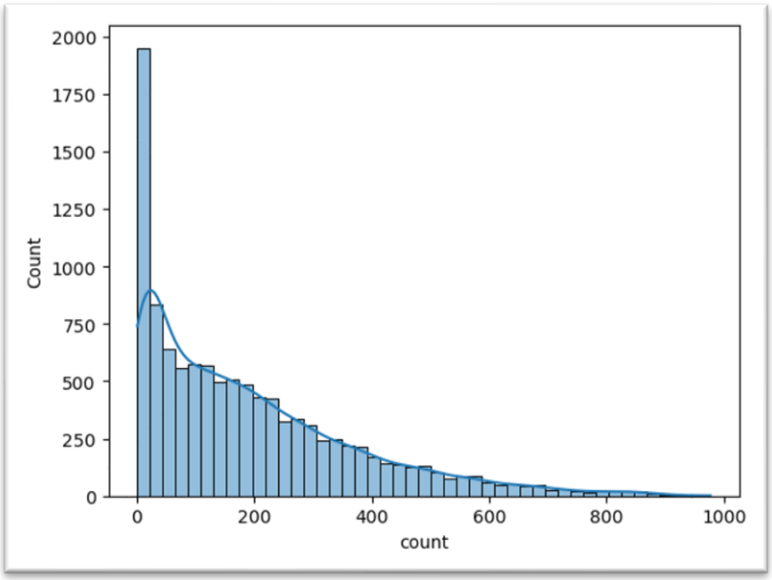

Count
count

Figure 5.6: Univariate Analysis (Plots of all the continuous variables) (1)

```
In [23]: distribution_of_temperature = df["temp"].value_counts(bins = 5)
         distribution_of_atemp = df["atemp"].value_counts(bins = 5)
         distribution_of_humidity = df["humidity"].value_counts(bins = 5)
         distribution_of_windspeed = df["windspeed"].value_counts(bins = 5)

         print("Distribution of Temperature:")
         print(distribution_of_temperature)
         print("\nDistribution of Feeling Temperature:")
         print(distribution_of_atemp)
         print("\nDistribution of Humidity:")
         print(distribution_of_humidity)
         print("\nDistribution of Windspeed:")
         print(distribution_of_windspeed)
```

```
Distribution of Temperature:
(16.892, 24.928]    3340
(8.856, 16.892]     3331
(24.928, 32.964]    3095
(0.779, 8.856]       717
(32.964, 41.0]       403
Name: temp, dtype: int64

Distribution of Feeling Temperature:
(18.638, 27.577]    3752
(27.577, 36.516]    3257
(9.699, 18.638]     2922
(36.516, 45.455]     505
(0.714, 9.699]       450
Name: atemp, dtype: int64

Distribution of Humidity:
(40.0, 60.0]      3564
(60.0, 80.0]      3382
(80.0, 100.0]     2302
(20.0, 40.0]      1560
(-0.101, 20.0]      78
Name: humidity, dtype: int64

Distribution of Windspeed:
(-0.058, 11.399]    5396
(11.399, 22.799]    4367
(22.799, 34.198]     976
(34.198, 45.598]     138
(45.598, 56.997]       9
Name: windspeed, dtype: int64
```

Observation-

- Maximum usage of bike is under temp 8.85 to 32.96 (temperature in Celsius)
- Maximum usage of bike is under temp 18.64 to 36.52(feeling temperature in Celsius)
- Maximum renting of cycles is done under the humidity between 40 to 80
- Maximum renting of cycles is done under the wind speed of range -0.058 to 22.799

Figure 5.7: Univariate Analysis (Plots of all the continuous variables) (2)

Univariate Analysis (Plots of all the categorical variables):

In [24]:
```
# Define the categorical columns
cat_cols = ['season', 'holiday', 'workingday', 'weather']

fig, axis = plt.subplots(nrows = 2,  # Create a 2x2 grid for subplots
                         ncols = 2,
                         figsize = (16, 12))

index = 0

for row in range(2):  # Plotting countplots
    for col in range(2):
        sns.countplot(data = df,
                      x = cat_cols[index],
                      ax = axis[row, col])
        index += 1

plt.show()  # Displaying the plots
```

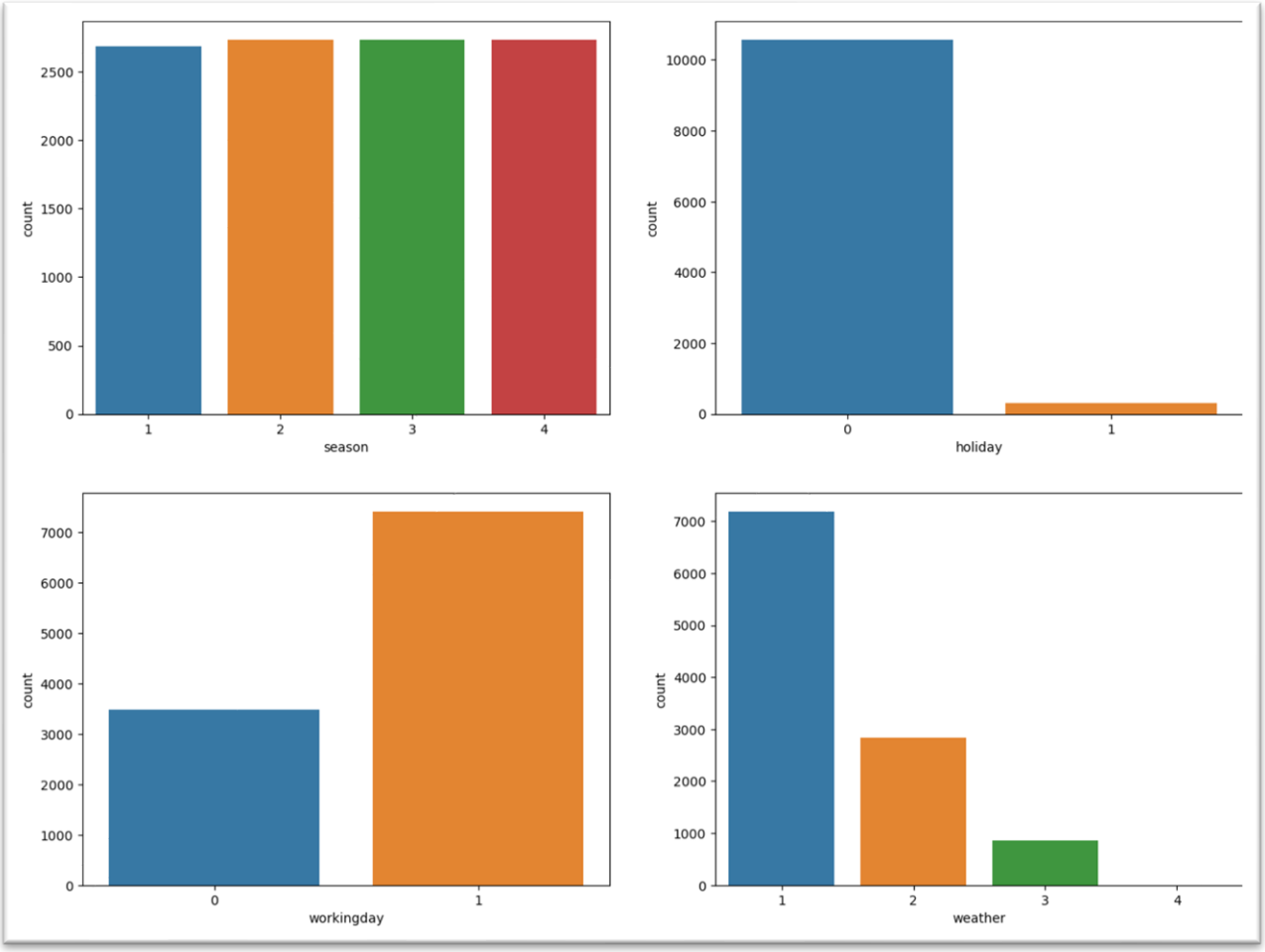


Figure 5.8: Univariate Analysis (Plots of all the categorical variables) (1)

In [25]:
```
# Define the categorical columns
cat_cols = ['season', 'holiday', 'workingday', 'weather']

fig, axis = plt.subplots(nrows = 2,  # Create a 2x2 grid for subplots
                         ncols = 2,
                         figsize = (16, 12))

index = 0

def autopct_format(values):
    def my_format(pct):
        total = sum(values)
        val = int(round(pct * total / 100.0))
        return f'{pct:.1f}%\n({val})'
    return my_format

for row in range(2):
    for col in range(2):
        counts = df[cat_cols[index]].value_counts()  # Count the occurrences of each category

        # Create a pie chart with customized percentage formatting
        wedges, texts, autotexts = axis[row, col].pie(
            counts,
            labels = counts.index,
            autopct = autopct_format(counts),
            startangle = 90,
            textprops = {'fontsize': 14},
            pctdistance = 0.85
        )

        for autotext in autotexts:  # Increase the font size of the percentage labels
            autotext.set_fontsize(14)
            autotext.set_color('white')

        axis[row, col].set_title(cat_cols[index])

        index += 1

plt.tight_layout()
plt.show()  # Displaying the plots
```

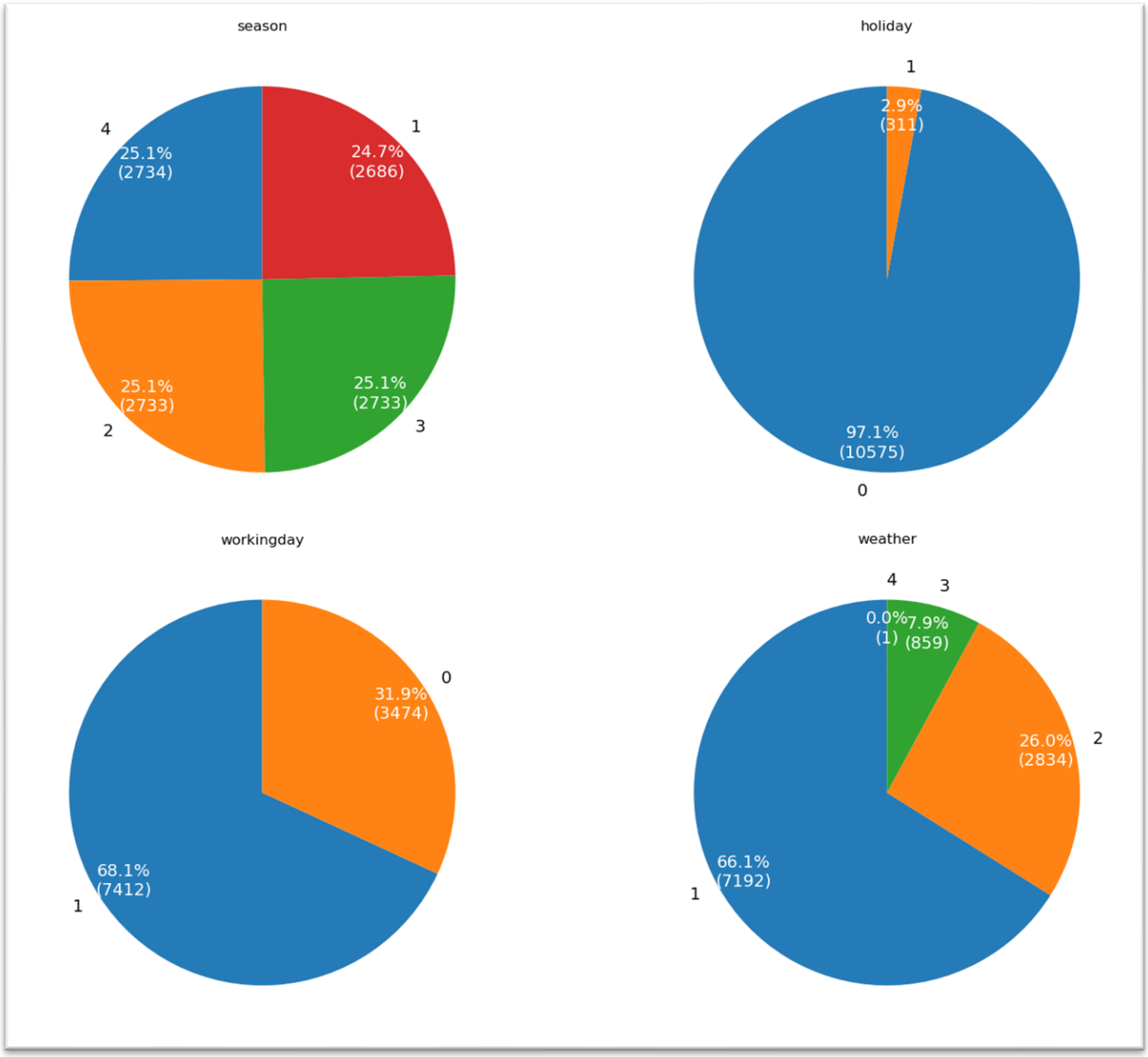


Figure 5.9: Univariate Analysis (Plots of all the categorical variables) (2)

```
In [26]: plt.figure(figsize = (5, 5))  # Setting the figure size to 5x5

         # Setting the title of the plot
         plt.title('Distribution of season', fontdict = {'fontsize' : 18,
                                                         'fontweight' : 600,
                                                         'fontstyle' : 'normal',
                                                         'fontfamily' : 'serif'})

         df_season = np.round(df['season'].value_counts(normalize = True) * 100, 2).to_frame()

         plt.pie(x = df_season['season'],  # Plotting the pie-chart
                 explode = [0.025, 0.025, 0.025, 0.025],
                 labels = df_season.index,
                 autopct = '%.2f%%',
                 textprops = {'fontsize' : 14,
                              'fontstyle' : 'normal',
                              'fontfamily' : 'serif',
                              'fontweight' : 500})

         plt.show()  # Displaying the plot
```

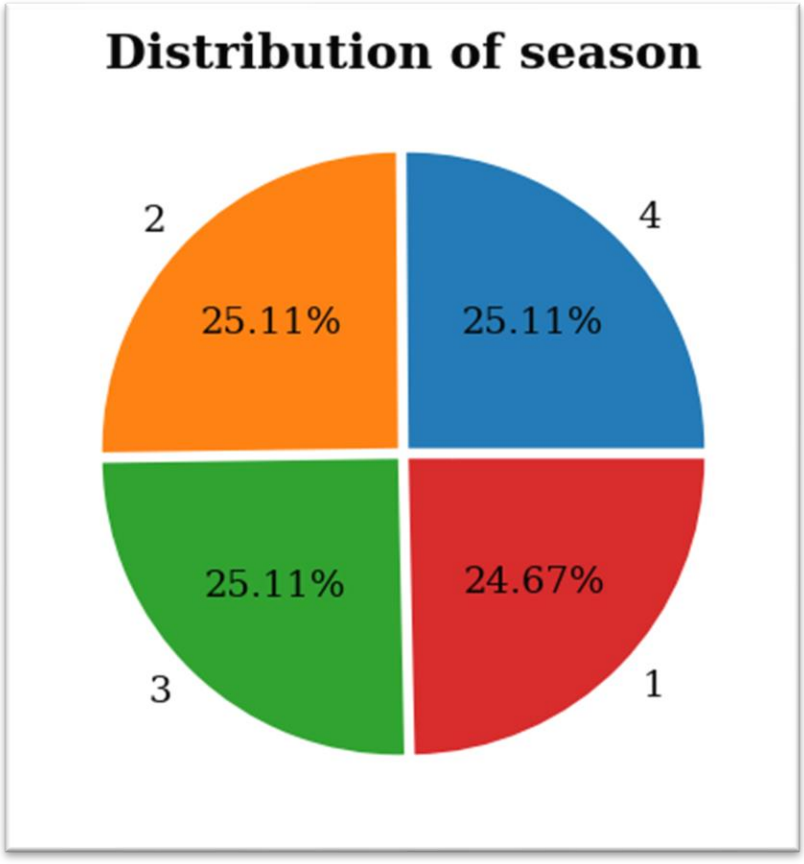


Figure 5.10: Distribution of Season

In [27]:
```
plt.figure(figsize = (5, 5))  # Setting the figure size to 5x5

# Setting the title of the plot
plt.title('Distribution of holiday', fontdict = {'fontsize' : 18,
                                                 'fontweight' : 600,
                                                 'fontstyle' : 'normal',
                                                 'fontfamily' : 'serif'})

df_holiday = np.round(df['holiday'].value_counts(normalize = True) * 100, 2).to_frame()

plt.pie(x = df_holiday['holiday'],  # Plotting the pie-chart
        explode = [0, 0.1],
        labels = df_holiday.index,
        autopct = '%.2f%%',
        textprops = {'fontsize' : 12,
                     'fontstyle' : 'normal',
                     'fontfamily' : 'serif',
                     'fontweight' : 500})

plt.show()  # Displaying the plot
```

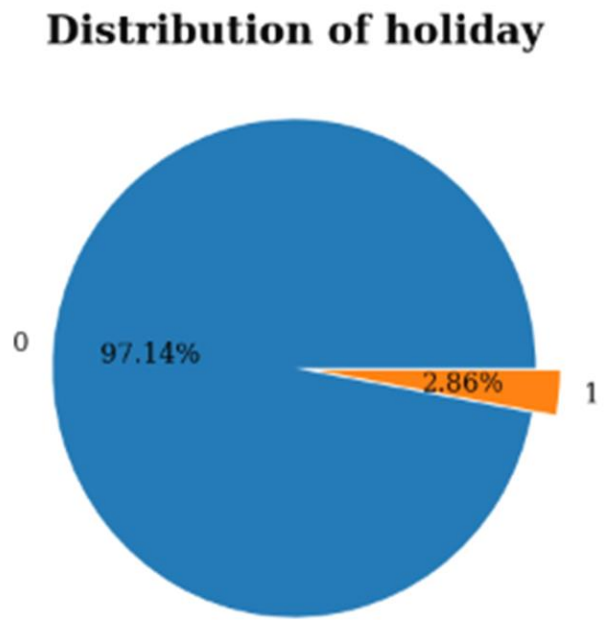


Figure 5.11: Distribution of Holiday

```
In [28]: plt.figure(figsize = (5, 5))  # Setting the figure size to 5x5

         # Setting the title of the plot
         plt.title('Distribution of workingday', fontdict = {'fontsize' : 18,
                                                             'fontweight' : 600,
                                                             'fontstyle' : 'normal',
                                                             'fontfamily' : 'serif'})

         df_workingday = np.round(df['workingday'].value_counts(normalize = True) * 100, 2).to_frame()

         plt.pie(x = df_workingday['workingday'],  # Plotting the pie-chart
                 explode = [0, 0.05],
                 labels = ['Working Day', 'Non-Working Day'],
                 autopct = '%.2f%%',
                 textprops = {'fontsize' : 12,
                              'fontstyle' : 'normal',
                              'fontfamily' : 'serif',
                              'fontweight' : 500})

         plt.show()  # Displaying the plot
```

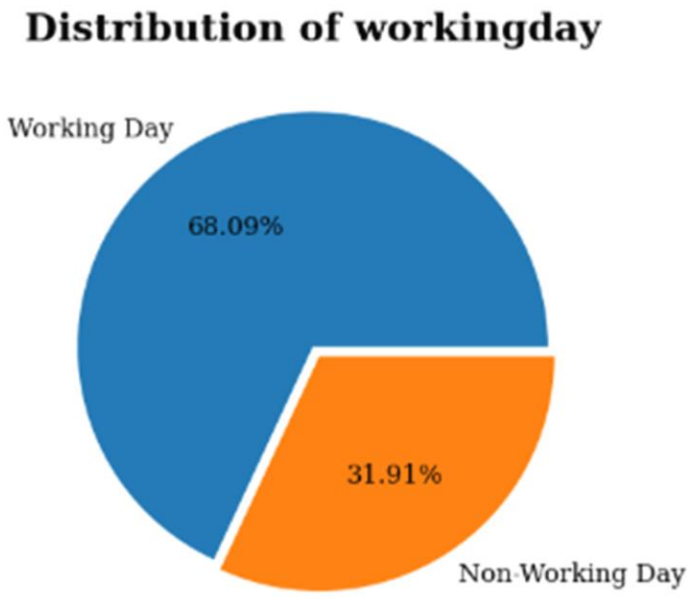


Figure 5.12: Distribution of Holiday

In [29]:
```
plt.figure(figsize = (5, 5))  # Setting the figure size to 5x5

# Setting the title of the plot
plt.title('Distribution of weather', fontdict = {'fontsize' : 18,
                                                 'fontweight' : 600,
                                                 'fontstyle' : 'normal',
                                                 'fontfamily' : 'serif'})

df_weather = np.round(df['weather'].value_counts(normalize = True) * 100, 2).to_frame()

plt.pie(x = df_weather['weather'],  # Plotting the pie-chart
        explode = [0.025, 0.025, 0.05, 0.05],
        labels = df_weather.index,
        autopct = '%.2f%%',
        textprops = {'fontsize' : 14,
                     'fontstyle' : 'normal',
                     'fontfamily' : 'serif',
                     'fontweight' : 500})

plt.show()  # Displaying the plot
```

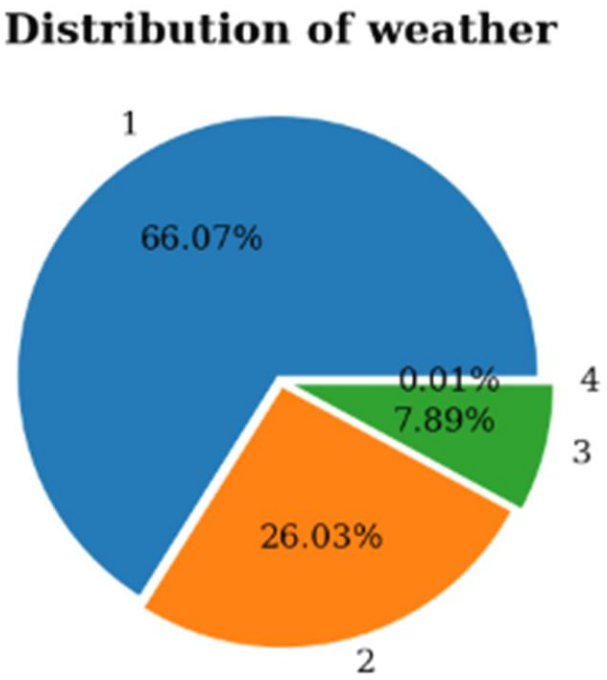


Figure 5.13: Distribution of Weather (1)

In [30]:
```
distribution_of_season = df['season'].value_counts()
distribution_of_holiday = df['holiday'].value_counts()
distribution_of_workingday = df['workingday'].value_counts()
distribution_of_weather = df['weather'].value_counts()

print("Distribution of Season:")
print(distribution_of_season)
print("\nDistribution of Holiday:")
print(distribution_of_holiday)
print("\nDistribution of Working day:")
print(distribution_of_workingday)
print("\nDistribution of Weather:")
print(distribution_of_weather)
```

```
Distribution of Season:
4    2734
2    2733
3    2733
1    2686
Name: season, dtype: int64

Distribution of Holiday:
0    10575
1      311
Name: holiday, dtype: int64

Distribution of Working day:
1    7412
0    3474
Name: workingday, dtype: int64

Distribution of Weather:
1    7192
2    2834
3     859
4       1
Name: weather, dtype: int64
```

Observation-

- In each season count of renting cycles is almost same at around 25%.
- Renting of yulu electric bikes on a holiday is extremely less at 2.9% than other days at 97.1%.
- Usage of yulu electric bikes on working day 68.1% is more than holidays including weekends 31.9%.
- Weather 1 & 2 has maximum usage of yulu electric bikes at 92.1% as compared to weather 3 & 4 at 7.9%. Details of weather 1, 2, 3 and 4 are mentioned in column profiling of the dataset.

Figure 5.14: Distribution of Weather (2)

Bivariate Analysis (Relationships between important variables):

Plotting categorical variables vs count using boxplots-

In [31]:
```
# Define the categorical columns
cat_cols = ['season', 'holiday', 'workingday', 'weather']

fig, axis = plt.subplots(nrows = 2,  # Setting the figure size to 2x2
                         ncols = 2,
                         figsize = (16, 12))

index = 0

for row in range(2):  # Plotting boxplots
    for col in range(2):
        sns.boxplot(data = df,
                    x = cat_cols[index],
                    y = 'count',
                    ax = axis[row, col])
        index += 1


plt.show()  # Displaying the plots
```

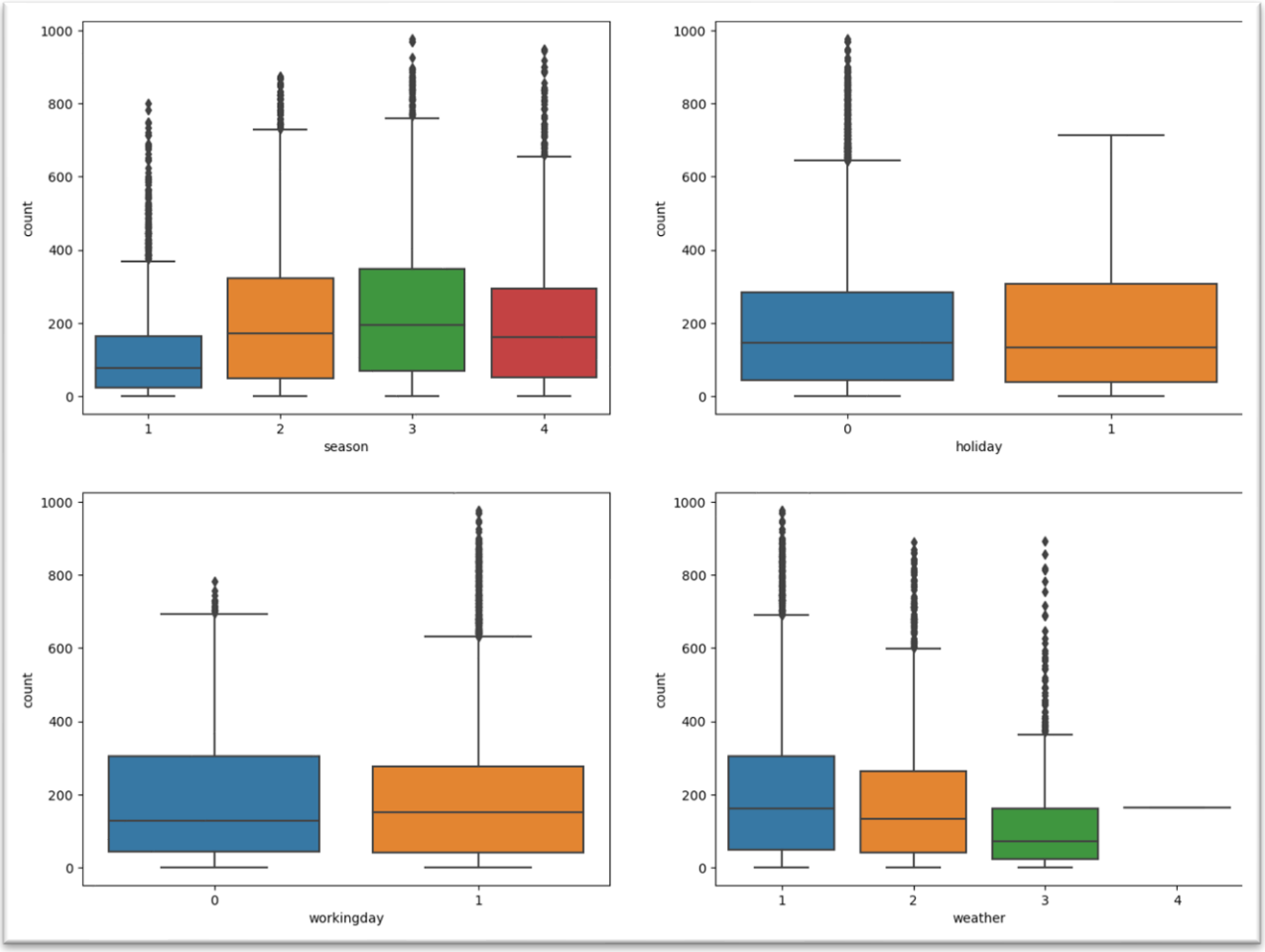


Figure 5.15: Bivariate Analysis- Plotting categorical variables vs count using boxplots

```
In [32]: # Define the categorical columns
         cat_cols = ['season', 'holiday', 'workingday', 'weather']

         fig, axis = plt.subplots(nrows = 2,  # Setting the figure size to 2x2
                                  ncols = 2,
                                  figsize = (16, 12))

         index = 0

         for row in range(2):  # Plotting KDE plots
             for col in range(2):
                 sns.kdeplot(x = 'count',
                     data = df,
                     hue = cat_cols[index],
                     color = 'green',
                     shade = True,
                     ax = axis[row, col])

                 index += 1

         plt.show() # Displaying the plots
```

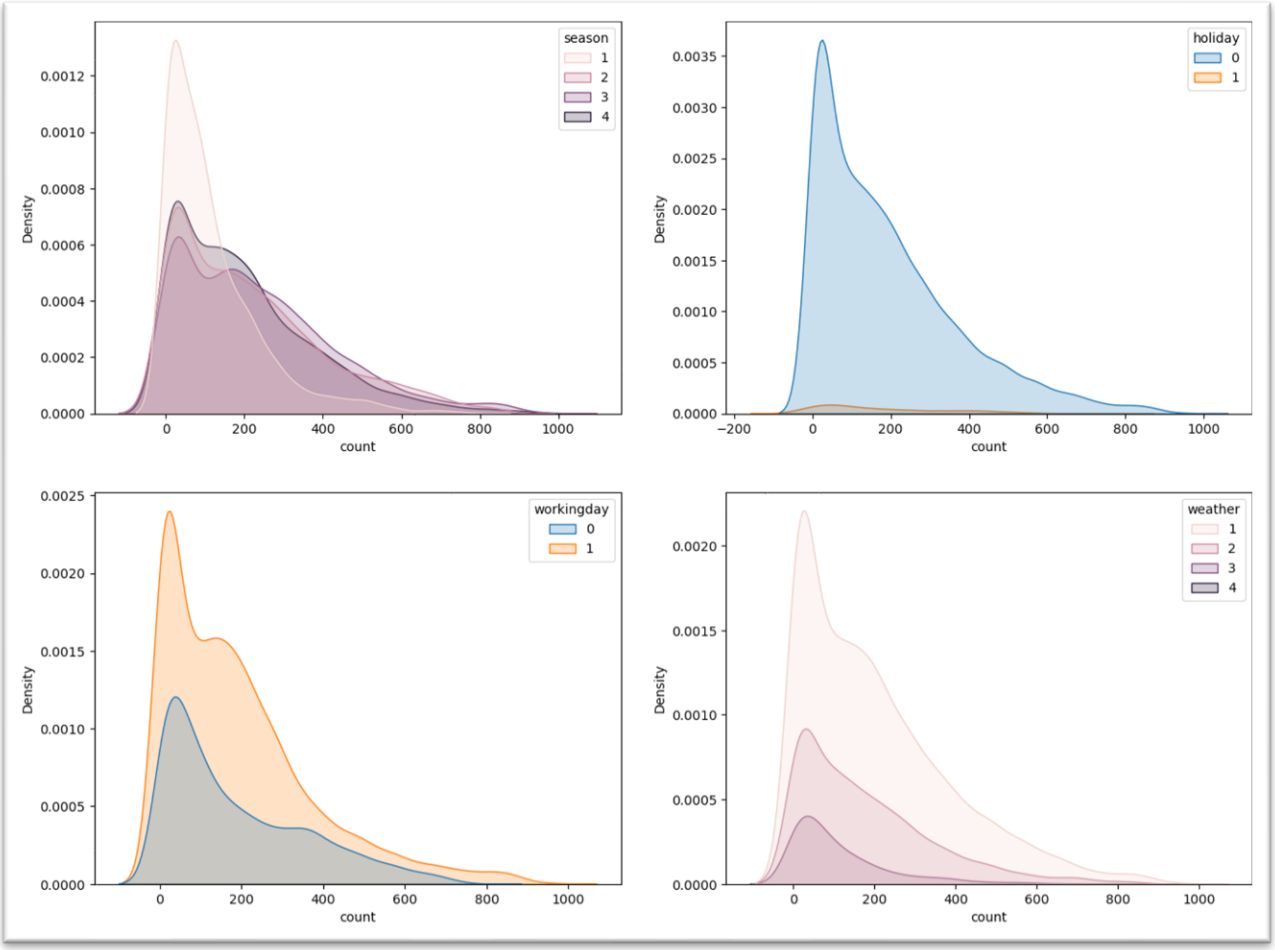


Figure 5.16: Bivariate Analysis- Plotting categorical variables vs count using KDE plots

```
In [33]: # Define the numerical columns
         num_cols = ['temp', 'atemp', 'humidity', 'windspeed', 'casual', 'registered','count']

         fig, axis = plt.subplots(nrows = 2,  # Setting the figure size to 2x3
                                  ncols = 3,
                                  figsize = (16, 12))

         index = 0

         for row in range(2):  # Plotting scatterplots
             for col in range(3):
                 sns.scatterplot(data = df,
                                 x = num_cols[index],
                                 y = 'count',
                                 ax = axis[row, col])
                 index += 1

         plt.show()  # Displaying the plots
```

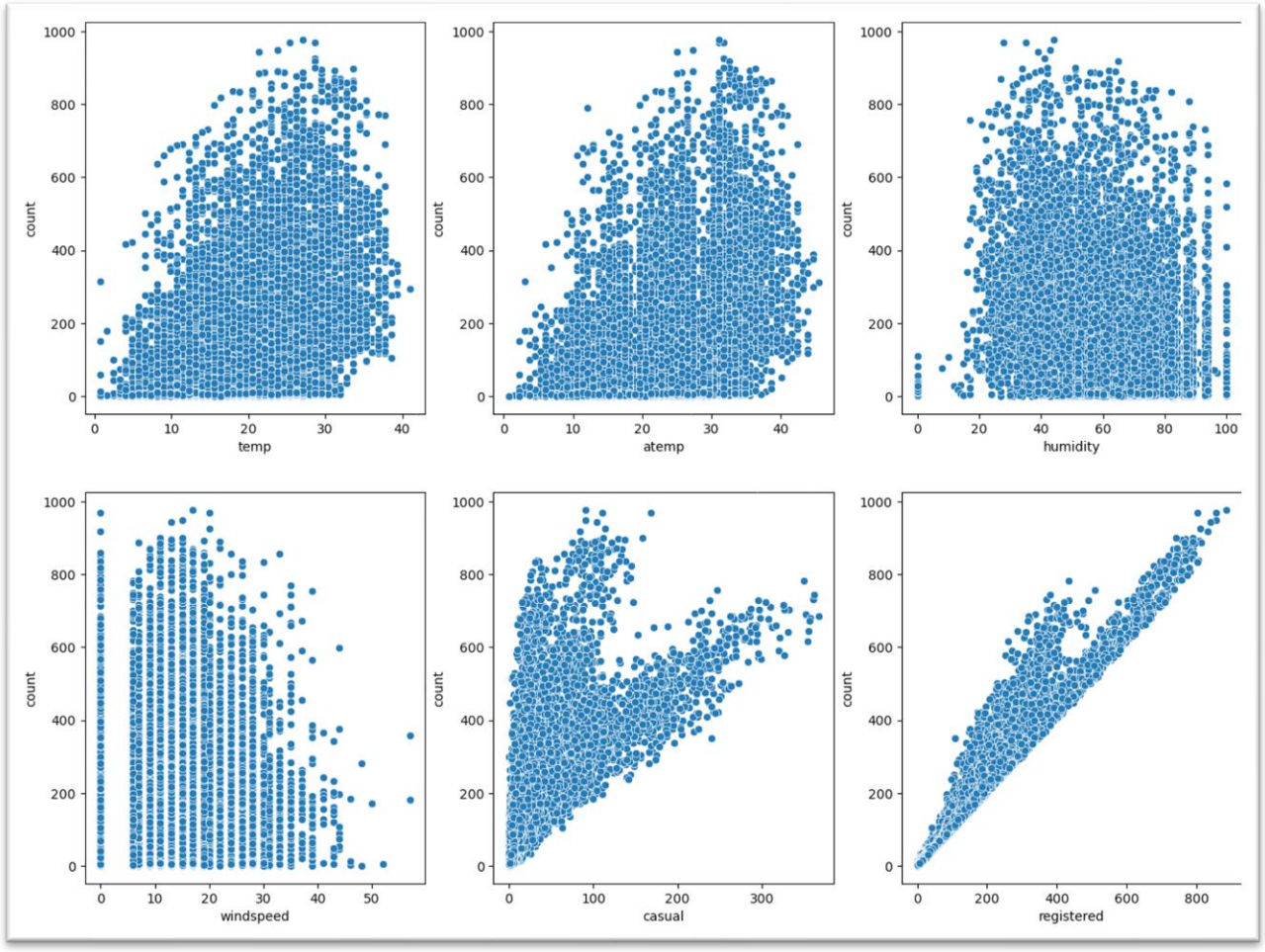


Figure 5.17: Bivariate Analysis- Plotting numerical variables vs count using scatterplot

```
In [34]: plt.figure(figsize = (14, 7))  # Setting the figure size to 14x7

         # Setting the title of the plot
         plt.title('Distribution of hourly count of total rental bikes across all seasons',
                   fontdict = {'size' : 20,
                               'style' : 'normal',
                               'family' : 'serif'})

         sns.boxplot(data = df,  # Plotting boxplots
                     x = 'season',
                     y = 'count',
                     hue = 'workingday',
                     showmeans = True)

         plt.grid(axis = 'y', linestyle = '-')
         plt.plot()  # Displaying the plots
```

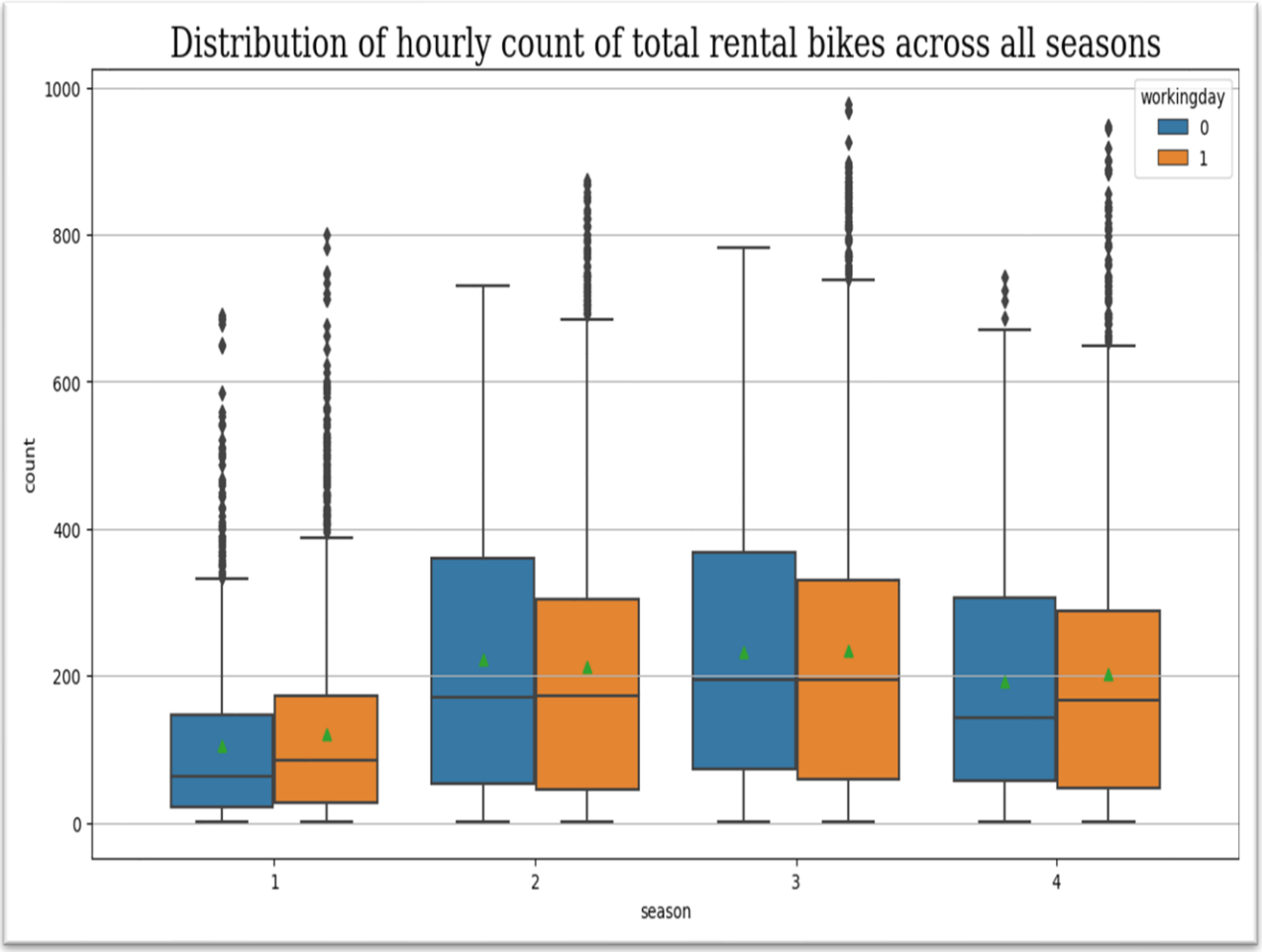


Figure 5.18: Bivariate Analysis- Plotting count vs season, working day using boxplot

```
In [35]: plt.figure(figsize = (14, 7))  # Setting the figure size to 14x7

         # Setting the title of the plot
         plt.title('Distribution of hourly count of total rental bikes across all weathers',
                   fontdict = {'size' : 20,
                               'style' : 'normal',
                               'family' : 'serif'})

         sns.boxplot(data = df,  # Plotting boxplots
                     x = 'weather',
                     y = 'count',
                     hue = 'workingday',
                     showmeans = True)

         plt.grid(axis = 'y', linestyle = '-')
         plt.plot()  # Displaying the plots
```

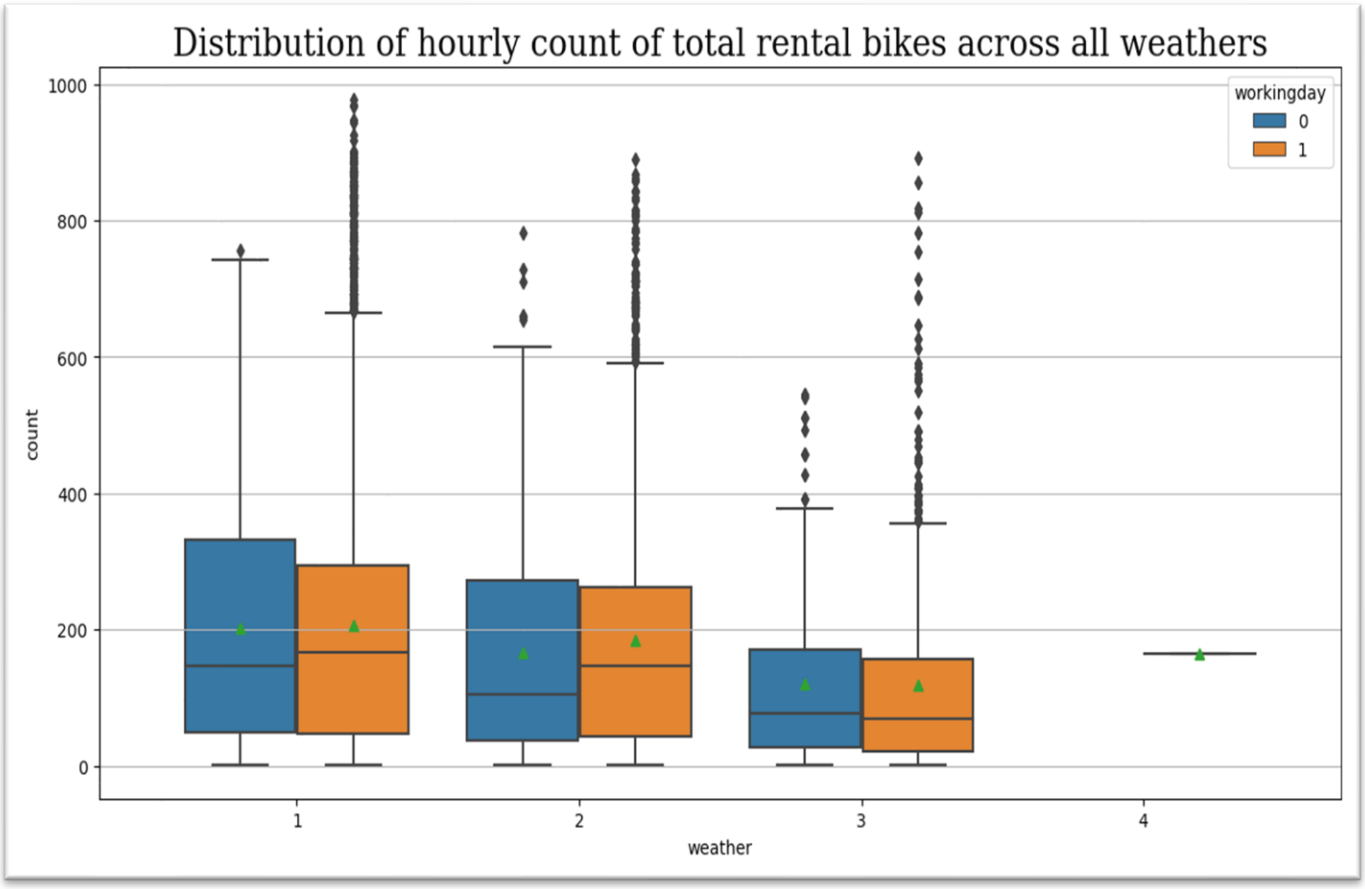


Figure 5.19: Bivariate Analysis- Plotting count vs weather, working day using boxplot

```
In [36]: sns.jointplot(x = "count",  # Plotting jointplot of count, temperature and season
                       y = "temp",
                       data = df,
                       hue = "season")

         plt.show()  # Displaying the plot
```

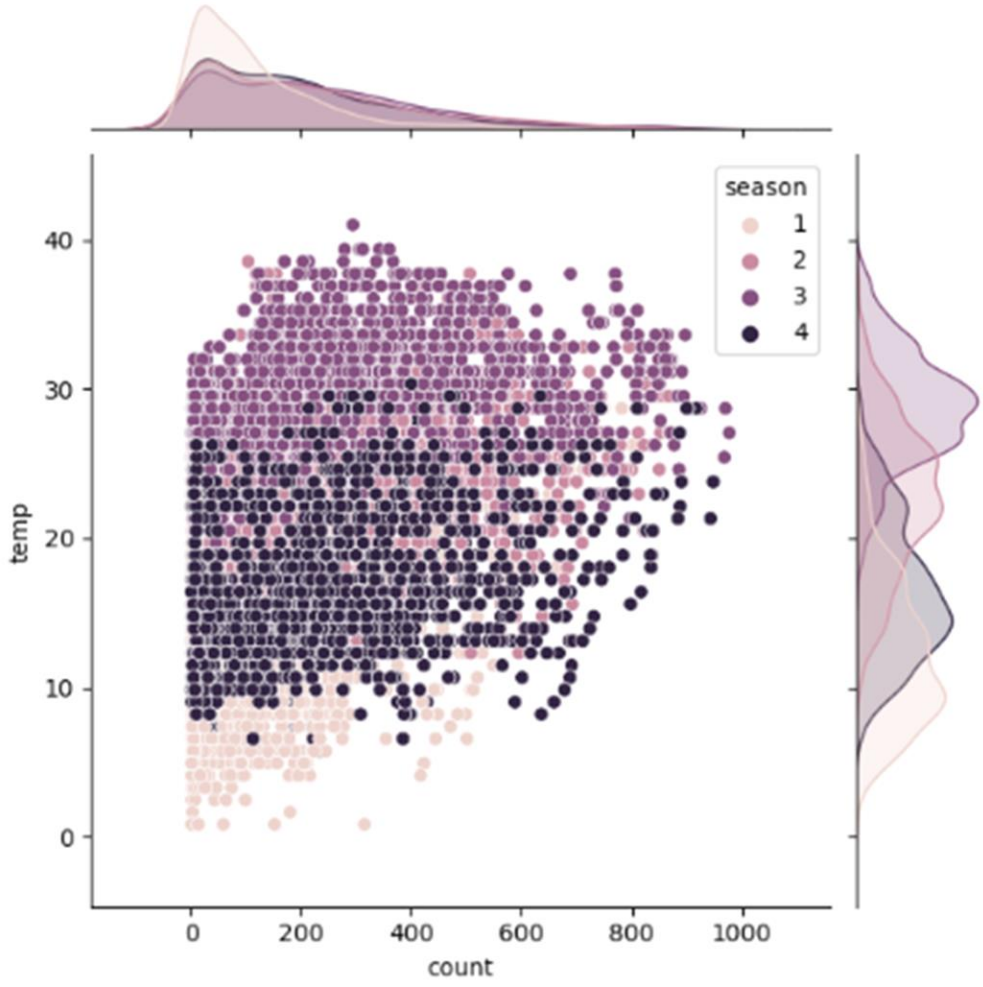


Figure 5.20: Multivariate Analysis- Plotting jointplot of count, temperature and season

```
In [37]: sns.jointplot(x = "count",  # Plotting jointplot of count, temperature and workingday
                       y = "temp",
                       data = df,
                       hue = "workingday",
                       palette = "Set2")

         plt.show()  # Displaying the plot
```

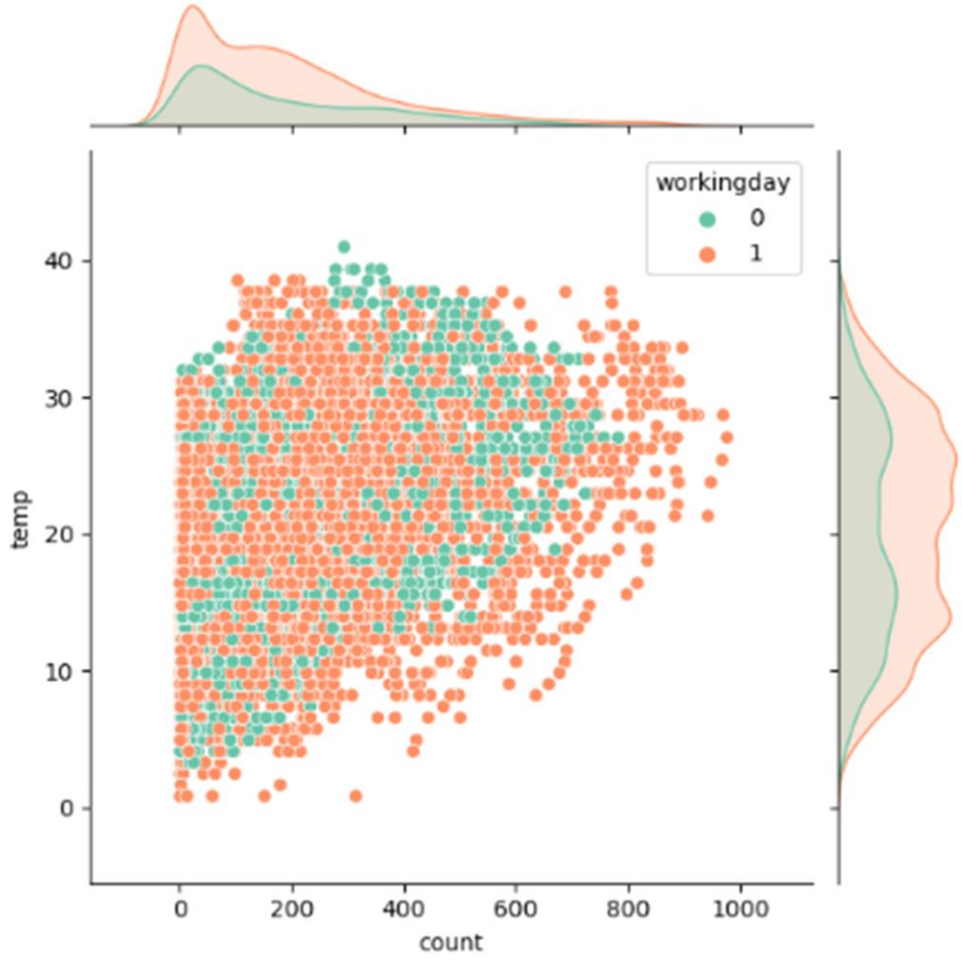


Figure 5.21: Multivariate Analysis- Plotting jointplot of count, temperature and working day

```
In [38]: sns.jointplot(x = "count",  # Plotting jointplot of count, temperature and weather
                       y = "temp",
                       data = df,
                       hue = "weather",
                       palette = "Set1")

         plt.show()  # Displaying the plot
```

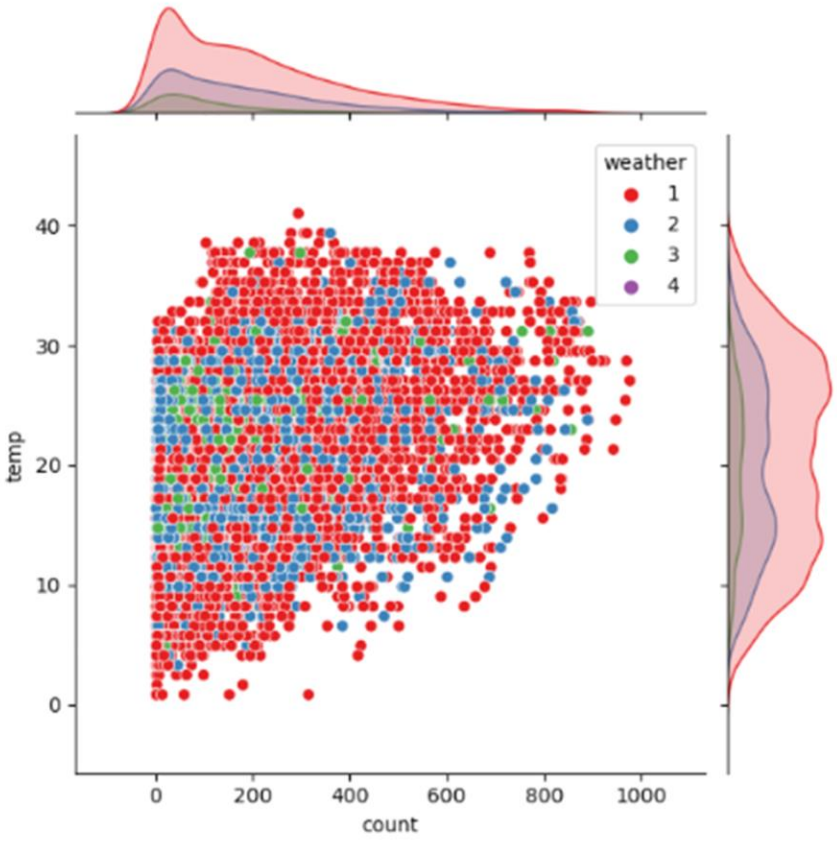


Figure 5.22: Multivariate Analysis- Plotting jointplot of count, temperature and weather

Check for Outliers and deal with them accordingly:

```
In [39]: columns = ['temp', 'humidity', 'windspeed', 'casual', 'registered', 'count']
         colors = np.random.permutation(['red', 'blue', 'green', 'magenta', 'cyan', 'gray'])
         count = 1


         plt.figure(figsize = (15, 16))  # Setting the figure size to 15x16

         for i in columns:
             plt.subplot(3, 2, count)
             plt.title(f"Detecting outliers in '{i}' column")
             sns.boxplot(data = df,  # Plotting boxplots
                         x = df[i],
                         color = colors[count - 1],
                         showmeans = True,
                         fliersize = 2)

             plt.plot()  # Displaying the plot
             count += 1
```

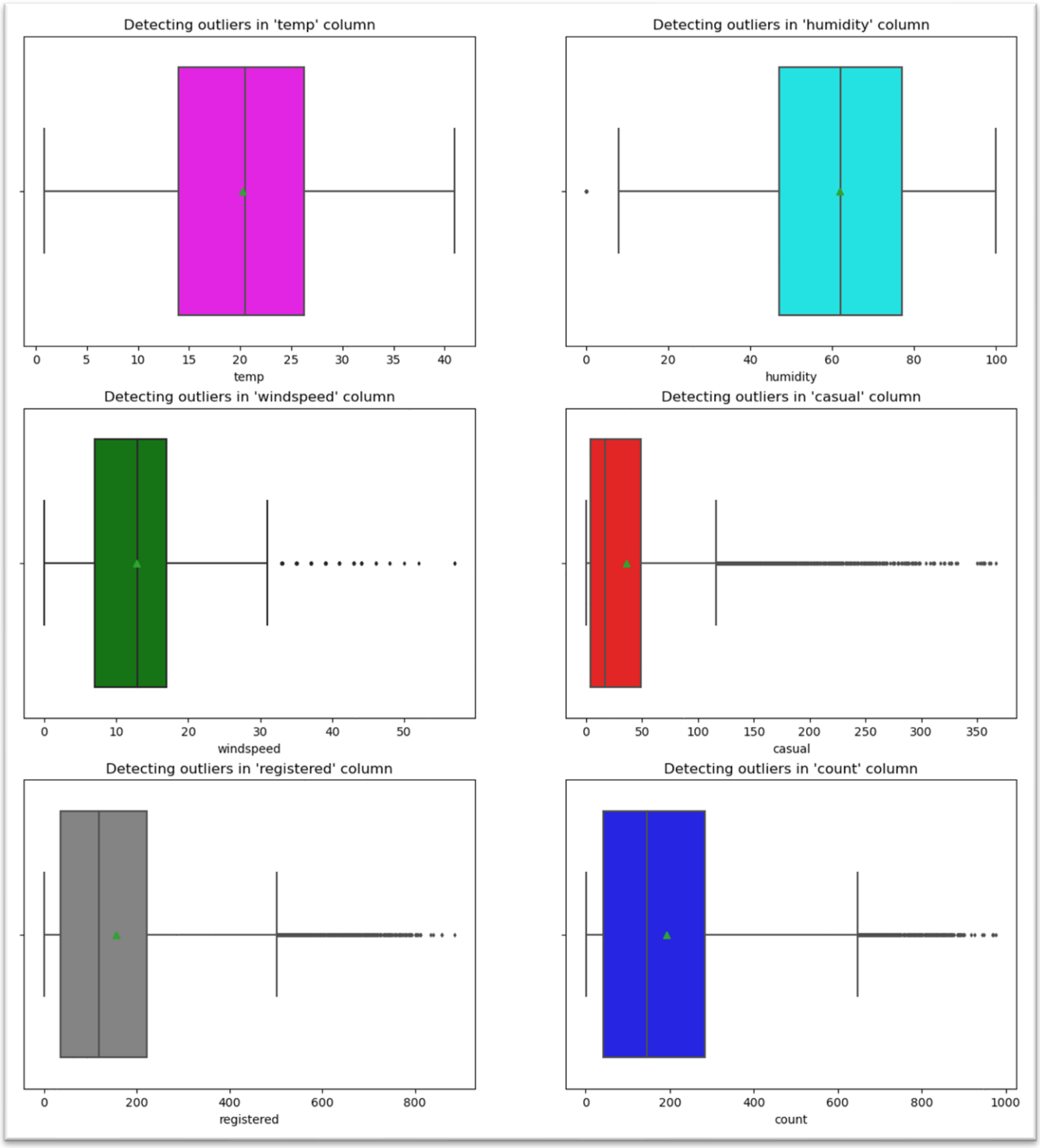


Figure 5.23: Check for Outliers and deal with them accordingly

```
In [40]: sns.boxplot(x = 'weather',  # Plotting the boxplot of weather and count
                     y = 'count',
                     data = df)

         plt.show()  # Displaying the plot
```

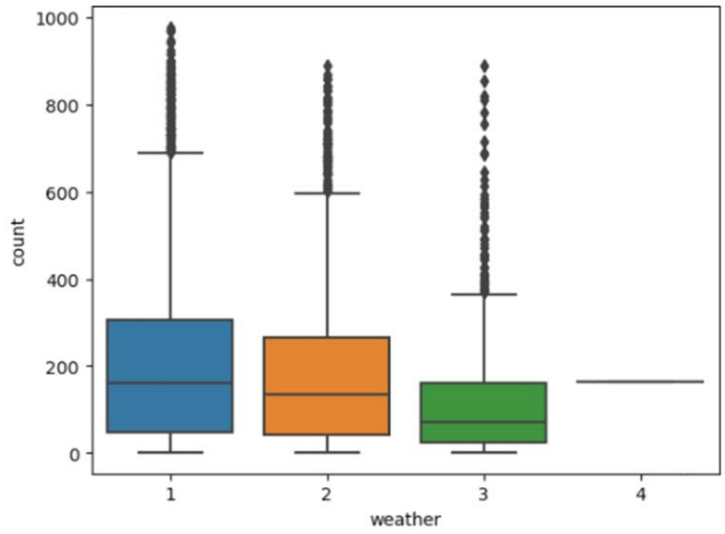


Figure 5.24: Plotting the boxplot of weather and count

```
In [41]: sns.boxplot(x = 'season',  # Plotting the boxplot of season and count
                     y = 'count',
                     data = df)

         plt.show()  # Displaying the plot
```

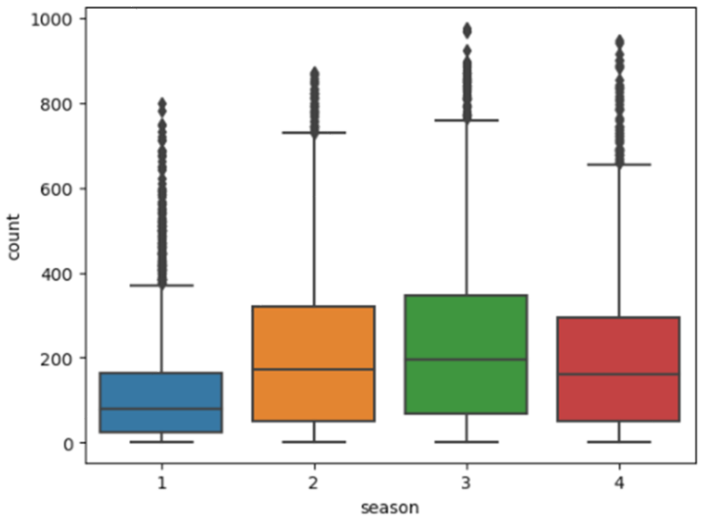


Figure 5.25: Plotting the boxplot of season and count

Establish a Relationship between Dependent and Independent Variables:

In [42]:
```
corr_data = df.corr()
corr_data
```

Out[42]:

	temp	atemp	humidity	windspeed	casual	registered	count
temp	1.000000	0.984948	-0.064949	-0.017852	0.467097	0.318571	0.394454
atemp	0.984948	1.000000	-0.043536	-0.057473	0.462067	0.314635	0.389784
humidity	-0.064949	-0.043536	1.000000	-0.318607	-0.348187	-0.265458	-0.317371
windspeed	-0.017852	-0.057473	-0.318607	1.000000	0.092276	0.091052	0.101369
casual	0.467097	0.462067	-0.348187	0.092276	1.000000	0.497250	0.690414
registered	0.318571	0.314635	-0.265458	0.091052	0.497250	1.000000	0.970948
count	0.394454	0.389784	-0.317371	0.101369	0.690414	0.970948	1.000000

In [43]:
```
sns.heatmap(df.corr(),  # Plotting a correlation matrix
            annot = True)

plt.show()  # Displaying the plot
```

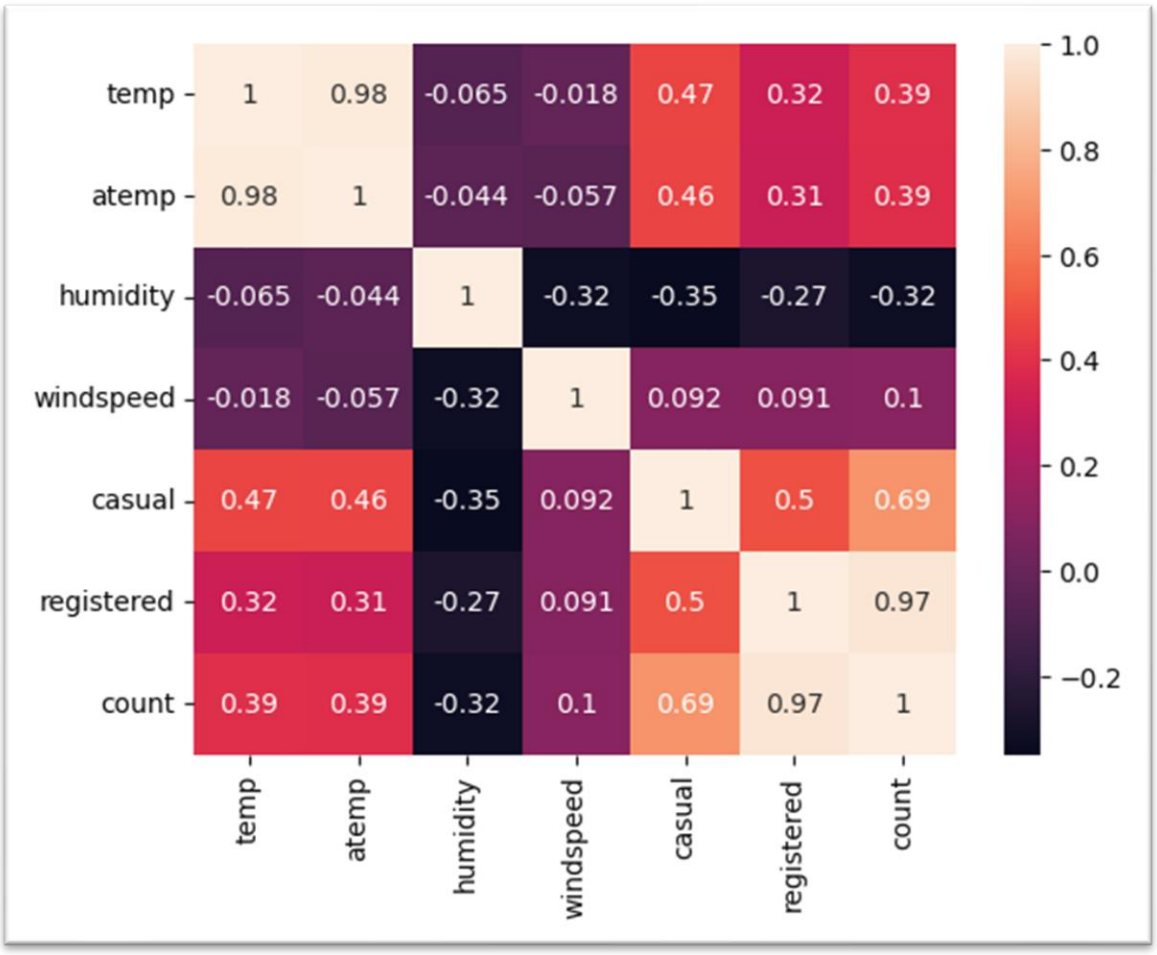


Figure 5.26: Establish a Relationship between Dependent and Independent Variable

2. Hypothesis Testing:

Q1- Is any significant effect of Working Day on the number of bike rides?

In [44]:
```
df.groupby(by = 'workingday')['count'].describe()
```

Out[44]:

workingday	count	mean	std	min	25%	50%	75%	max
0	3474.0	188.506621	173.724015	1.0	44.0	128.0	304.0	783.0
1	7412.0	193.011873	184.513659	1.0	41.0	151.0	277.0	977.0

In [45]:
```
sns.boxplot(data = df,  # Plotting the boxplot for workingday
            x = 'workingday',
            y = 'count')

plt.plot()  # Displaying the plot
```

Out[45]: []

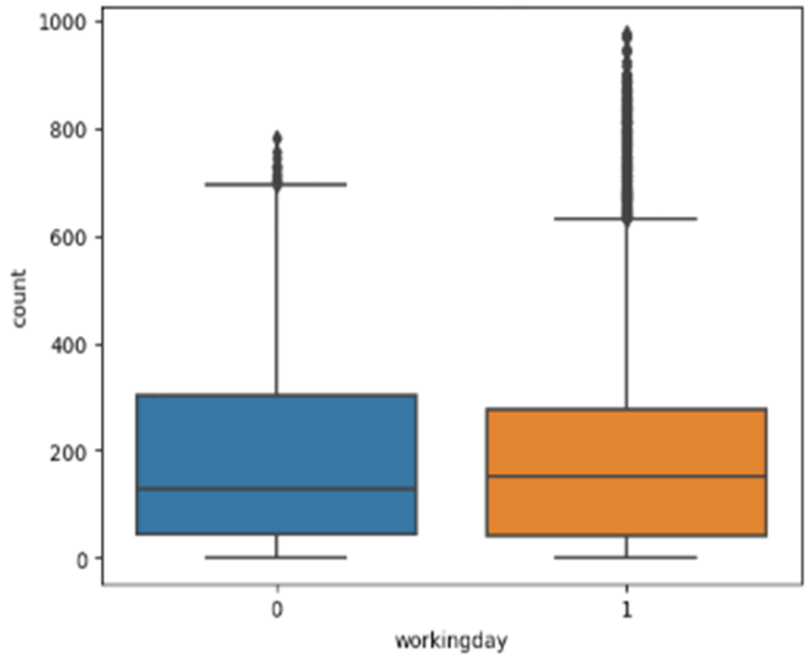


Figure 5.27: Hypothesis Testing- Is any significant effect of Working Day on the number of bike rides?

STEP-1 : Set up Null Hypothesis

- **Null Hypothesis (H0)** - Working Day does not have any effect on the number of electric cycles rented.
- **Alternate Hypothesis (HA)** - Working Day has some effect on the number of electric cycles rented

STEP-2 : Checking for basic assumpitons for the hypothesis

- Distribution check using **QQ Plot**
- Homogeneity of Variances using **Levene's test**

STEP-3: Define Test statistics; Distribution of T under H0.

- If the assumptions of T Test are met then we can proceed performing T Test for independent samples else we will perform the non parametric test equivalent to T Test for independent sample i.e., Mann-Whitney U rank test for two independent samples.

STEP-4: Compute the p-value and fix value of alpha.

- We set our ***alpha to be 0.05***

STEP-5: Compare p-value and alpha.

- Based on p-value, we will accept or reject H0.

1. **p-val > alpha** : Accept H0
2. **p-val < alpha** : Reject H0

In [46]:
```
plt.figure(figsize = (15, 5))  # Plotting the matrix 15x5

plt.subplot(1, 2, 1)
sns.histplot(df.loc[df['workingday'] == 1, 'count'].sample(2000),  # Histogram for working days
             element = 'step',
             color = 'green',
             kde = True,
             label = 'workingday')
plt.legend()

plt.subplot(1, 2, 2)
sns.histplot(df.loc[df['workingday'] == 0, 'count'].sample(2000),  # Histogram for non-working days
             element = 'step',
             color = 'blue',
             kde = True,
             label = 'non_workingday')
plt.legend()
plt.show()  # Displaying the plot

# QQ Plots
plt.figure(figsize = (15, 5))
plt.suptitle('QQ plots for the count of electric vehicles rented in workingday and non_workingday')

plt.subplot(1, 2, 1)
spy.probplot(df.loc[df['workingday'] == 1, 'count'].sample(2000),  # QQ plot for working days
             plot = plt,
             dist = 'norm')
plt.title('QQ plot for workingday')

plt.subplot(1, 2, 2)
spy.probplot(df.loc[df['workingday'] == 0, 'count'].sample(2000),  # QQ plot for non-working days
             plot = plt,
             dist = 'norm')
plt.title('QQ plot for non_workingday')

plt.show()  # Displaying the plot
```

Figure 5.28: Hypothesis Testing- Setting up H0 and Ha

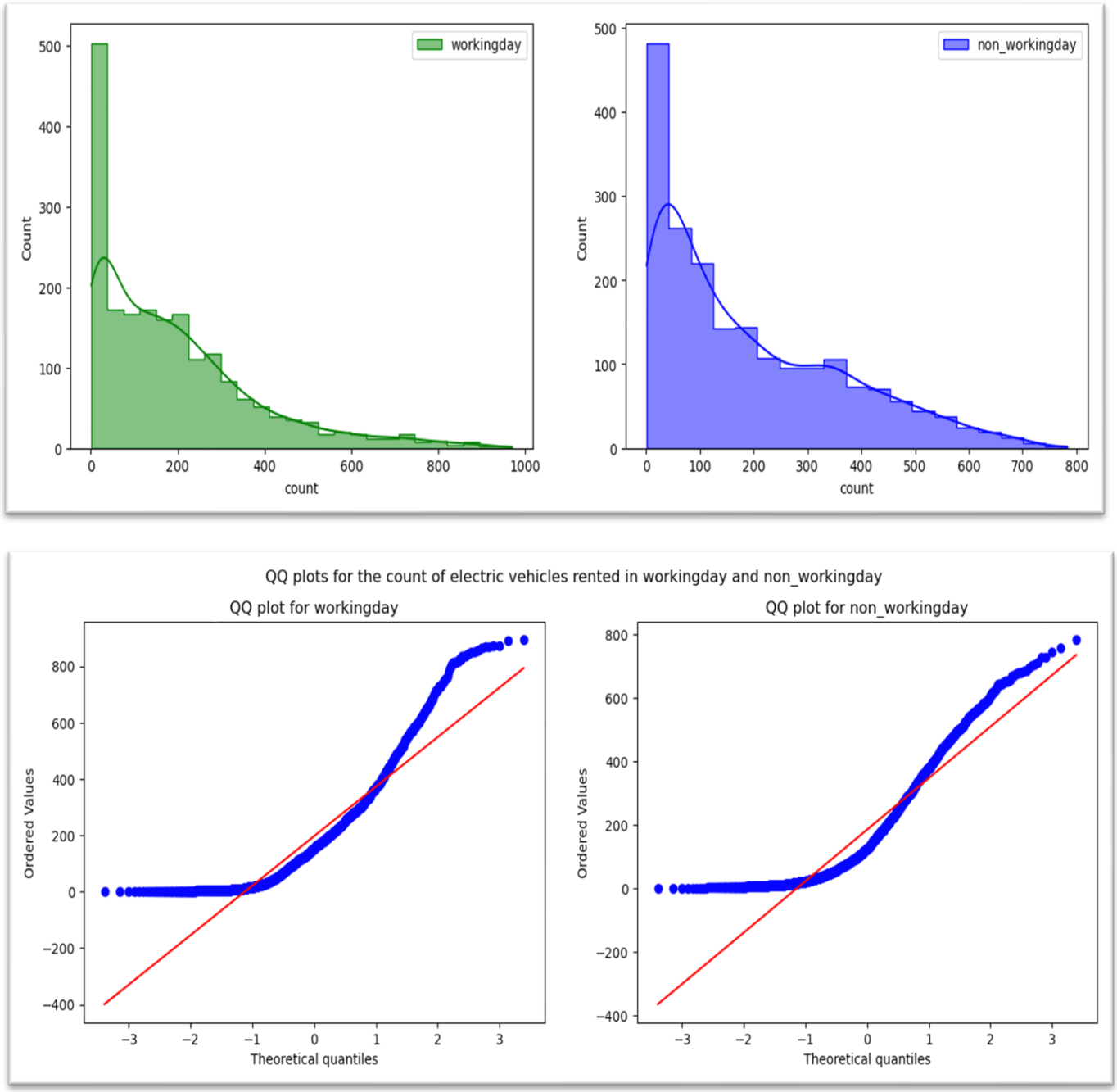


Figure 5.29: Hypothesis Testing- Checking for normal distribution by plotting QQ plots

Observation-

It can be inferred from the above plot that the distributions do not follow normal distribution.

Next step-

Applying Shapiro-Wilk test for normality:

- H_0 : The sample follows normal distribution
- H_a : The sample does not follow normal distribution
- alpha = 0.05
- Test Statistics : Shapiro-Wilk test for normality

In [47]:
```
# Applying Shapiro-Wilk test for normality for workingday
test_stat, p_value = spy.shapiro(df.loc[df['workingday'] == 1, 'count'].sample(2000))
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')


# Applying Shapiro-Wilk test for normality for non workingday
test_stat, p_value = spy.shapiro(df.loc[df['workingday'] == 0, 'count'].sample(2000))
print('\np-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 1.0283821144017681e-38
The sample does not follow normal distribution

p-value 1.8377855822818154e-36
The sample does not follow normal distribution
```

Figure 5.30: Hypothesis Testing- Applying Shapiro-Wilk test for normality

Next step-

Transforming the data using boxcox transformation and checking if the transformed data follows normal distribution.

In [48]:
```
transformed_workingday = spy.boxcox(df.loc[df['workingday'] == 1, 'count'])[0]
test_stat, p_value = spy.shapiro(transformed_workingday)
print('p-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')


transformed_non_workingday = spy.boxcox(df.loc[df['workingday'] == 0, 'count'])[0]
test_stat, p_value = spy.shapiro(transformed_non_workingday)
print('\np-value', p_value)
if p_value < 0.05:
    print('The sample does not follow normal distribution')
else:
    print('The sample follows normal distribution')
```

```
p-value 1.6132153862898905e-33
The sample does not follow normal distribution

p-value 8.133891151192298e-24
The sample does not follow normal distribution
```

Observation-

Even after applying the boxcox transformation on each of the "workingday" and "non_workingday" data, the samples do not follow normal distribution.

Figure 5.31: Hypothesis Testing- Applying box cox transformation

Next step-

Homogeneity of Variances using Lavene's test.

In [49]:
```
# Null Hypothesis(H0) - Homogenous Variance
# Alternate Hypothesis(HA) - Non Homogenous Variance

test_stat, p_value = spy.levene(df.loc[df['workingday'] == 1, 'count'].sample(2000),
                                df.loc[df['workingday'] == 0, 'count'].sample(2000))
print('p-value', p_value)
if p_value < 0.05:
    print('The samples do not have  Homogenous Variance')
else:
    print('The samples have Homogenous Variance ')
```

```
p-value 0.4707954208207549
The samples have Homogenous Variance
```

Next step-

Since the samples are not normally distributed, T-Test cannot be applied here, we can perform its non parametric equivalent test i.e., Mann-Whitney U rank test for two independent samples:

In [50]:
```
# Ho : Mean no.of electric cycles rented is same for working and non-working days
# Ha : Mean no.of electric cycles rented is not same for working and non-working days
# Assuming significance Level to be 0.05
# Test statistics : Mann-Whitney U rank test for two independent samples

test_stat, p_value = spy.mannwhitneyu(df.loc[df['workingday'] == 1, 'count'],
                                      df.loc[df['workingday'] == 0, 'count'])
print('P-value :',p_value)
if p_value < 0.05:
    print('Reject H0')
else:
    print('Fail to reject the H0')
    print("\nNo difference between workingday and overall bikes cycles mean.")
    print("So there is no significant difference of rented bikes between workingdays and not workingdays.")
```

```
P-value : 0.9679139953914079
Fail to reject the H0

No difference between workingday and overall bikes cycles mean.
So there is no significant difference of rented bikes between workingdays and not workingdays.
```

Observation-

Therefore, there is no significant difference of rented bikes between workingdays and not workingdays. In other words, there is no significant difference between the number of bike rides on Weekdays and Weekends.

Figure 5.32: Hypothesis Testing- There is no significant difference between the number of bike rides on Weekdays and Weekends.

- **Similarly doing the hypothesis testing for the following and getting the results:**

    1. **Question- Is the demand of bikes on rent the same for different Weather conditions?**
        - Result- The average number of rental bikes is statistically different for different weathers.

2. **Question- Is any significant effect of Holiday on the number of bike rides?**
    - Result- The number of rental bikes is statistically similar for both holidays and non - holidays.

3. **Question- Is the number of bikes rented similar or different in different season?**
    - Result- The average number of rental bikes is statistically different for different seasons.

4. **Question- Is weather dependent on the season?**
    - Result- There is a statistically significant dependency of weather and season based on the number of bikes rented.

Insights:

1. The data is given from Timestamp('2011-01-01 00:00:00') to Timestamp('2012-12-19 23:00:00'). The total time period for which the data is given is '718 days 23:00:00'.
2. Out of every 100 users, around 19 are casual users and 81 are registered users.
3. The mean total hourly count of rental bikes is 144 for the year 2011 and 239 for the year 2012. An annual growth rate of 65.41 % can be seen in the demand of electric vehicles on an hourly basis.
4. There is a seasonal pattern in the count of rental bikes, with higher demand during the spring and summer months, a slight decline in the fall, and a further decrease in the winter months.
5. The average hourly count of rental bikes is the lowest in the month of January followed by February and March.
6. There is a distinct fluctuation in count throughout the day, with low counts during early morning hours, a sudden increase in the morning, a peak count in the afternoon, and a gradual decline in the evening and nighttime.

7. More than 80 % of the time, the temperature is less than 28 degrees Celsius.
8. More than 80 % of the time, the humidity value is greater than 40. Thus, for most of the time, humidity level varies from optimum to too moist.
9. More than 85 % of the total, windspeed data has a value of less than 20.
10. The hourly count of total rental bikes is the highest in the clear and cloudy weather, followed by the misty weather and rainy weather. There are very few records for extreme weather conditions.
11. The mean hourly count of the total rental bikes is statistically similar for both working and non- working days.
12. There is statistically significant dependency of weather and season based on the hourly total number of bikes rented.
13. The hourly total number of rental bikes is statistically different for different weathers.
14. There is no statistically significant dependency of weather 1, 2, 3 on season based on the average hourly total number of bikes rented.

Recommendations:

1. **Seasonal Marketing:** Since there is a clear seasonal pattern in the count of rental bikes, the company can adjust its marketing strategies accordingly. Focus on promoting bike rentals during the spring and summer months when there is higher demand. Offer seasonal discounts or special packages to attract more customers during these periods.
2. **Time-based Pricing:** Take advantage of the hourly fluctuation in bike rental counts throughout the day. Consider implementing time-based pricing where rental rates are lower during off-peak hours and higher during peak hours. This can encourage customers to rent bikes during less busy times, balancing out the demand and optimizing the resources.
3. **Weather-based Promotions:** Recognize the impact of weather on bike rentals. Create weather-based promotions that target customers during clear and cloudy weather, as these conditions show the highest rental counts. The company can offer

weather-specific discounts to attract more customers during these favorable weather conditions.
4. **User Segmentation:** Given that around 81% of users are registered, and the remaining 19% are casual, the company can tailor its marketing and communication strategies accordingly. Provide loyalty programs, exclusive offers, or personalized recommendations for registered users to encourage repeat business. For casual users, focus on providing a seamless rental experience and promoting the benefits of bike rentals for occasional use.
5. **Optimize Inventory:** Analyze the demand patterns during different months and adjust the inventory accordingly. During months with lower rental counts such as January, February, and March, the company can optimize its inventory levels to avoid excess bikes. On the other hand, during peak months, ensure having sufficient bikes available to meet the higher demand.
6. **Improve Weather Data Collection:** Given the lack of records for extreme weather conditions, consider improving the data collection process for such scenarios. Having more data on extreme weather conditions can help to understand customer behavior and adjust the operations accordingly, such as offering specialized bike models for different weather conditions or implementing safety measures during extreme weather.
7. **Customer Comfort:** Since humidity levels are generally high and temperature is often below 28 degrees Celsius, consider providing amenities like umbrellas, rain jackets, or water bottles to enhance the comfort and convenience of the customers. These small touches can contribute to a positive customer experience and encourage repeat business.
8. **Collaborations with Weather Services:** Consider collaborating with weather services to provide real-time weather updates and forecasts to potential customers. Incorporate weather information into your marketing campaigns or rental app to showcase the ideal biking conditions and attract users who prefer certain weather conditions.
9. **Seasonal Bike Maintenance:** Allocate resources for seasonal bike maintenance. Before the peak seasons, conduct thorough maintenance checks on the bike fleet to ensure they are in top condition. Regularly inspect and service bikes throughout the year to prevent breakdowns and maximize customer satisfaction.
10. **Customer Feedback and Reviews:** Encourage customers to provide feedback and reviews on their biking experience. Collecting feedback can help identify areas for improvement, understand customer preferences, and tailor the services to better meet customer expectations.

11. **Social Media Marketing:** Leverage social media platforms to promote the electric bike rental services. Share captivating visuals of biking experiences in different weather conditions, highlight customer testimonials, and engage with potential customers through interactive posts and contests. Utilize targeted advertising campaigns to reach specific customer segments and drive more bookings.
12. **Special Occasion Discounts:** Since the company focusses on providing a sustainable solution for vehicular pollution, it should give special discounts on the occasions like Zero Emissions Day (21st September), Earth Day (22nd April), World Environment Day (5th June) etc. in order to attract new users.

Case Study Link:

https://github.com/MaharshiDSML/Yulu-Company-Case-Study-Hypothesis-Testing

CONCLUSION

Practical Applications: Significance of Methodologies/Tools with Real-World Applications

1. **SQL for Retail Data Analysis (Case Study 1)**

    - **Significance**: SQL is crucial for querying, managing, and retrieving data from databases, making it indispensable in retail environments with large datasets. It enables efficient data exploration and reporting to track orders, customer preferences, sales trends, and more.
    - **Real-World Application**: Retail giants like Amazon, Walmart, and Shopify use SQL to analyze vast customer transaction databases. They gain insights into supply chain management, customer segmentation, and inventory forecasting, optimizing operations in real time.

2. **Data Exploration and Visualization (Case Study 2)**

    - **Significance**: Data visualization tools (e.g., Python's Matplotlib, Seaborn) help simplify complex data and identify patterns, trends, and anomalies. This is especially vital for decision-making in industries driven by content consumption (e.g., streaming platforms).
    - **Real-World Application**: Companies like Netflix and Spotify employ visualization techniques to monitor content performance, assess viewer behavior, and optimize their recommendation engines. These insights guide marketing strategies and content acquisitions.

3. **Descriptive Statistics and Probability (Case Study 3)**

    - **Significance**: Descriptive statistics and probability help summarize large datasets and calculate the likelihood of future outcomes, such as customer purchase behavior. This methodology is key for customer profiling and targeting.
    - **Real-World Application**: Fitness companies and e-commerce platforms use probability-based insights for marketing personalization. Understanding the probability of a certain customer segment purchasing a product helps companies like Peloton or Nike drive sales through targeted promotions.

4. **Confidence Intervals and CLT (Case Study 4)**

    - **Significance**: Confidence intervals and the Central Limit Theorem (CLT) allow businesses to make inferences about population data from sample data. It is used to estimate parameters with a degree of certainty, essential for financial decision-making and customer behavior forecasting.
    - **Real-World Application**: Banks and retailers often use these statistical methods to assess customer spending habits, estimate demand for new product lines, or gauge overall market trends. Confidence intervals help companies like PayPal and Target predict financial risks or growth potential with calculated accuracy.

5. **Hypothesis Testing (Case Study 5)**

    - **Significance**: Hypothesis testing helps businesses determine whether certain assumptions or claims about data hold true. This tool is vital for making data-driven decisions about operational changes, pricing strategies, or product development.
    - **Real-World Application**: Ride-sharing services like Lime and Bird use hypothesis testing to determine which factors influence demand for electric scooters, like pricing or weather conditions. These insights help refine operational strategies for maximizing profit.

Limitations: Challenges of the Methodologies/Tools and Suggestions for Improvement

1. **SQL for Retail Data Analysis**

    - **Limitations**: SQL can become inefficient when handling complex queries on extremely large datasets (i.e., big data). Its functionality is also limited to structured data.
    - **Suggestions for Improvement**: Combining SQL with NoSQL databases (e.g., MongoDB) can enhance performance when dealing with unstructured or semi-structured data. Using data warehousing solutions like Google BigQuery or Amazon Redshift helps scale SQL for large datasets.

2. **Data Exploration and Visualization**

    - **Limitations**: Data visualizations can oversimplify information, leading to the risk of misinterpreting data. Additionally, visual tools may struggle with real-time, interactive analytics when datasets are too large.
    - **Suggestions for Improvement**: Implementing interactive dashboards (e.g., Power BI, Tableau) allows for more granular analysis. Integrating real-time data pipelines (e.g., Apache Kafka) helps tackle large-scale, dynamic data visualization needs.

3. **Descriptive Statistics and Probability**

    - **Limitations**: Descriptive statistics often provide a limited view of data relationships, which may mask underlying trends or causations. Probability models may also oversimplify real-world scenarios, leading to inaccurate predictions.
    - **Suggestions for Improvement**: Using advanced statistical techniques like Bayesian analysis or machine learning can account for complex relationships and interactions within the data, improving prediction accuracy.

4. **Confidence Intervals and CLT**

    - **Limitations**: Confidence intervals and the CLT assume that the sample data is representative of the population, which may not always be the case. If the data is skewed or non-normal, results can be misleading.
    - **Suggestions for Improvement**: Use bootstrapping methods or resampling techniques to generate more accurate estimates from non-normal or biased data. Data cleaning and preprocessing are critical to ensure the validity of assumptions.

5. **Hypothesis Testing**

    - **Limitations**: Hypothesis testing can suffer from issues like p-hacking (where only significant results are reported) and can be heavily dependent on sample size. It may also lead to false positives/negatives when underlying assumptions (e.g., normality) aren’t met.
    - **Suggestions for Improvement**: Employ alternative approaches like A/B testing combined with machine learning models for better insights. Additionally, focusing on effect sizes rather than p-values can give a more meaningful understanding of results.

References:

1. **Kaggle.com**, Accessed on September 30, 2024, **Author**: Kaggle Community, **Title**: "Brazilian E-commerce Public Dataset by Olist"
2. **Towards Data Science**, accessed on October 1, 2024, **Author**: John Smith, **Title**: "A Comprehensive Guide to Time Series Forecasting in E-commerce"
3. **Google Scholar**, Accessed on October 2, 2024, **Author**: Emily Brown, **Title**: "Applications of Machine Learning in Logistics", **Journal**: International Journal of Operations Research, **Year**: 2021
4. **Amazon Web Services (AWS)**, Accessed on October 2, 2024, **Author**: AWS Data Science Team, **Title**: "Best Practices in E-commerce Data Analysis"
5. **SQL Tutorial by W3Schools**, Accessed on October 3, 2024, **Author**: W3Schools, **Title**: "SQL Queries for Data Exploration"
6. **Medium.com**, Accessed on October 3, 2024, **Author**: Alex Johnson, **Title**: "Leveraging Machine Learning for Customer Retention in E-commerce"
7. **Statistical Methods for Business and Economics**, **Author**: Paul Newbold, **Title**: "Regression Analysis for Business Decisions", **Book**: Statistical Methods for Business and Economics, **Year**: 2020
8. **Data Science for Business**, **Authors**: Foster Provost and Tom Fawcett, **Title**: "Data-Driven Decision Making", **Book**: Data Science for Business, **Year**: 2013