AIRLINE DATA ANALYSIS

Prakhar Tripathi

Ishita Mishra

DESCRIPTION

This is a data analysis project that is done by using MySQL and Python, in this analysis we have to

solve the business problem of the airlines, and also for solving it we have to
find some insights.

BUSINESS PROBLEM

There is a company that operates a diverse fleet of aircraft ranging from small business jets to

medium-sized machines. However, They are currently facing challenges due to several factors such as stricter

environmental regulations, higher flight taxes, increased interest rates, rising fuel prices, and a tight labor market resulting in increased labor costs.

As a result, the company's profitability is under pressure, and they are seeking ways to address this issue. To tackle this challenge, they are looking to analyze their database to find ways to increase their occupancy rate, which can help boost the average profit earned per seat.

MAIN GOAL OF THE ANALYSIS

1. Enhancing the occupancy of flight.

2. Maintain the pricing

3. Enhancing the customer experience

The main goal is to find the way to make the airline more profitable and also resolving the issues like occupancy rate pricing and customer satisfaction

TOOLS AND TECHNOLOGIES

SQLite3

SQLite3 for efficient querying and extraction of relevant data.

Python

Python for data manipulation, analysis, and visualization using the pandas library.

Matplotlib

Matplotlib for creating insightful visualizations to convey findings effectively.

Seaborn

Seaborn for creating insightful visualizations to convey findings effectively.

DATA EXPLORATION:

Importing the required packages

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3
```

Building Connection

```
conn = sqlite3.connect("C:\\Users\\ishita mishra\\Desktop\\Travel Project\\travel.sqlite")

cur = conn.cursor()
```

Dataset

```
cur.execute('select name from sqlite_master where type = "table";')

print("list of tables...")

t_list = [table[0] for table in cur.fetchall()]

print(t_list)
```

```
list of tables...
['aircrafts_data',
 'airports_data',
 'boarding_passes',
 'bookings',
 'flights',
 'seats',
 'ticket_flights',
 'tickets']
```

Checking the columns of all Tables

```
aircrafts_data = pd.read_sql_query("select * from aircrafts_data", conn)

print(aircrafts_data.columns)

print(aircrafts_data.head())
```

```
Index(['aircraft_code', 'model', 'range'], dtype='object')
```

	aircraft_code	model	range
0	773	{"en": "Boeing 777-300", "ru": "Боинг 777-300"}	11100
1	763	{"en": "Boeing 767-300", "ru": "Боинг 767-300"}	7900
2	SU9	{"en": "Sukhoi Superjet-100", "ru": "Сухой Суп...	3000
3	320	{"en": "Airbus A320-200", "ru": "Аэробус A320-...	5700
4	321	{"en": "Airbus A321-200", "ru": "Аэробус A321-...	5600

```
bookings = pd.read_sql_query("select * from bookings", conn)

print(bookings.columns)

print(bookings.head())
```

```
Index(['book_ref', 'book_date', 'total_amount'], dtype='object')
```

	book_ref	book_date	total_amount
0	00000F	2017-07-05 03:12:00+03	265700
1	000012	2017-07-14 09:02:00+03	37900
2	000068	2017-08-15 14:27:00+03	18100
3	000181	2017-08-10 13:28:00+03	131800
4	0002D8	2017-08-07 21:40:00+03	23600

```
airports_data = pd.read_sql_query("select * from airports_data", conn)

print(airports_data.columns)

print(airports_data.head())
```

```
Index(['airport_code', 'airport_name', 'city', 'coordinates', 'timezone'], dtype='object')
```

	airport_code	airport_name	city	coordinates	timezone
0	YKS	{"en": "Yakutsk Airport", "ru": "Якутск"}	{"en": "Yakutsk", "ru": "Якутск"}	(129.77099609375,62.0932998657226562)	Asia/Yakutsk
1	MJZ	{"en": "Mirny Airport", "ru": "Мирный"}	{"en": "Mirnyj", "ru": "Мирный"}	(114.03900146484375,62.534698486328125)	Asia/Yakutsk
2	KHV	{"en": "Khabarovsk-Novy Airport", "ru": "Хабар...	{"en": "Khabarovsk", "ru": "Хабаровск"}	(135.18800354004,48.5279998779300001)	Asia/Vladivostok
3	PKC	{"en": "Yelizovo Airport", "ru": "Елизово"}	{"en": "Petropavlovsk", "ru": "Петропавловск-К...	(158.453994750976562,53.1679000854492188)	Asia/Kamchatka
4	UUS	{"en": "Yuzhno-Sakhalinsk Airport", "ru": "Хом...	{"en": "Yuzhno-Sakhalinsk", "ru": "Южно-Сахали...	(142.718002319335938,46.8886985778808594)	Asia/Sakhalin

```
boarding_passes = pd.read_sql_query("select * from boarding_passes", conn)

# print(boarding_passes.columns)

# print(boarding_passes.head())
```

```
Index(['ticket_no', 'flight_id', 'boarding_no', 'seat_no'], dtype='object')
```

	ticket_no	flight_id	boarding_no	seat_no
0	0005435212351	30625	1	2D
1	0005435212386	30625	2	3G
2	0005435212381	30625	3	4H
3	0005432211370	30625	4	5D
4	0005435212357	30625	5	11A

```
flights = pd.read_sql_query("select * from flights", conn)
print(flights.columns)
print(flights.head())
```

```
Index(['flight_id', 'flight_no', 'scheduled_departure', 'scheduled_arrival',
       'departure_airport', 'arrival_airport', 'status', 'aircraft_code',
       'actual_departure', 'actual_arrival'],
      dtype='object')
```

	flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport	arrival_airport	status	aircraft_code	actual_departure	actual_arrival
0	1185	PG0134	2017-09-10 09:50:00+03	2017-09-10 14:55:00+03	DME	BTK	Scheduled	319	\N	\N
1	3979	PG0052	2017-08-25 14:50:00+03	2017-08-25 17:35:00+03	VKO	HMA	Scheduled	CR2	\N	\N
2	4739	PG0561	2017-09-05 12:30:00+03	2017-09-05 14:15:00+03	VKO	AER	Scheduled	763	\N	\N
3	5502	PG0529	2017-09-12 09:50:00+03	2017-09-12 11:20:00+03	SVO	UFA	Scheduled	763	\N	\N
4	6938	PG0461	2017-09-04 12:25:00+03	2017-09-04 13:20:00+03	SVO	ULV	Scheduled	SU9	\N	\N

```
seats = pd.read_sql_query("select * from seats", conn)
print(seats.columns)
print(seats.head())
```

```
Index(['aircraft_code', 'seat_no', 'fare_conditions'], dtype='object')
```

	aircraft_code	seat_no	fare_conditions
0	319	2A	Business
1	319	2C	Business
2	319	2D	Business
3	319	2F	Business
4	319	3A	Business

```
seats = pd.read_sql_query("select * from seats", conn)
print(seats.columns)
print(seats.head())
```

```
Index(['ticket_no', 'flight_id', 'fare_conditions', 'amount'], dtype='object')
```

	ticket_no	flight_id	fare_conditions	amount
0	0005432159776	30625	Business	42100
1	0005435212351	30625	Business	42100
2	0005435212386	30625	Business	42100
3	0005435212381	30625	Business	42100
4	0005432211370	30625	Business	42100

```
tickets = pd.read_sql_query("select * from tickets", conn)
print(tickets.columns)
print(tickets.head())
```

```
Index(['ticket_no', 'book_ref', 'passenger_id'], dtype='object')
```

	ticket_no	book_ref	passenger_id
0	0005432000987	06B046	8149 604011
1	0005432000988	06B046	8499 420203
2	0005432000989	E170C3	1011 752484
3	0005432000990	E170C3	4849 400049
4	0005432000991	F313DD	6615 976589

Checking for the Null Values in each table

```
for table in t_list:
    print("\n table: ",table)
    df_table = pd.read_sql_query(f"select * from {table}", conn)
    print(df_table.isnull().sum())
```

```
 table:  aircrafts_data
aircraft_code    0
model            0
range            0
dtype: int64

 table:  airports_data
airport_code    0
airport_name    0
city            0
coordinates     0
timezone        0
dtype: int64
```

```
 table:  boarding_passes
ticket_no      0
flight_id      0
boarding_no    0
seat_no        0
dtype: int64

 table:  bookings
book_ref        0
book_date       0
total_amount    0
dtype: int64
```

```
 table:  flights
flight_id              0
flight_no              0
scheduled_departure    0
scheduled_arrival      0
departure_airport      0
arrival_airport        0
status                 0
aircraft_code          0
actual_departure       0
actual_arrival         0
dtype: int64
```

```
 table:  seats
aircraft_code      0
seat_no            0
fare_conditions    0
dtype: int64

 table:  ticket_flights
ticket_no          0
flight_id          0
fare_conditions    0
amount             0
dtype: int64

 table:  tickets
ticket_no       0
book_ref        0
passenger_id    0
dtype: int64
```

EXPLORATORY DATA ANALYSIS

Q. PLANES THAT HAVE MORE THAN 100 SEATS

We went on a hunt to find the big airplanes with more than 100 seats. The SQL query unveils the top 6 aircraft, showcasing their dominance in seating capacity, ordered in descending fashion for a grand reveal.

Code:

```
seat_count = pd.read_sql_query("select aircraft_code, count(*) as Total_Seats from seats group by aircraft_code having count(*) > 100 order by Total_Seats desc", conn)

print(seat_count)
```

	aircraft_code	Total_Seats
0	773	402
1	763	222
2	321	170
3	320	140
4	733	130
5	319	116

Q. CHECKING THE FLIGHT STATUS COUNT AND ALSO CHECKING THE AIRCRAFT & AIRPORTS WHICH HAS MAXIMUM CANCELLATION.

It seems like there's a notable situation in our airline data adventure! We investigated flight statuses and found a whopping 414 cancellations tied to a specific aircraft departing from certain airports.

Code:

```
flight = pd.read_sql_query(f"SELECT status from flights", conn)

print(flight.value_counts())

Cancelled_aircraft = pd.read_sql_query("select aircraft_code,count(status) as seat_count from flights where status == 'Cancelled'", conn)

print(Cancelled_aircraft)

cancel_airport = pd.read_sql_query("SELECT departure_airport from flights where status == 'Cancelled'",conn)

print(cancel_airport.value_counts())
```

```
status
Arrived      16707
Scheduled    15383
On Time        518
Cancelled      414
Departed        58
Delayed         41
Name: count, dtype: int64
```

	aircraft_code	seat_count
0	CN1	414

```
departure_airport
DME                  31
SVO                  26
VKO                  20
LED                  15
OVB                  10
                     ..
MMK                   1
SLY                   1
NYA                   1
OGZ                   1
AAQ                   1
Name: count, Length: 100, dtype: int64
```

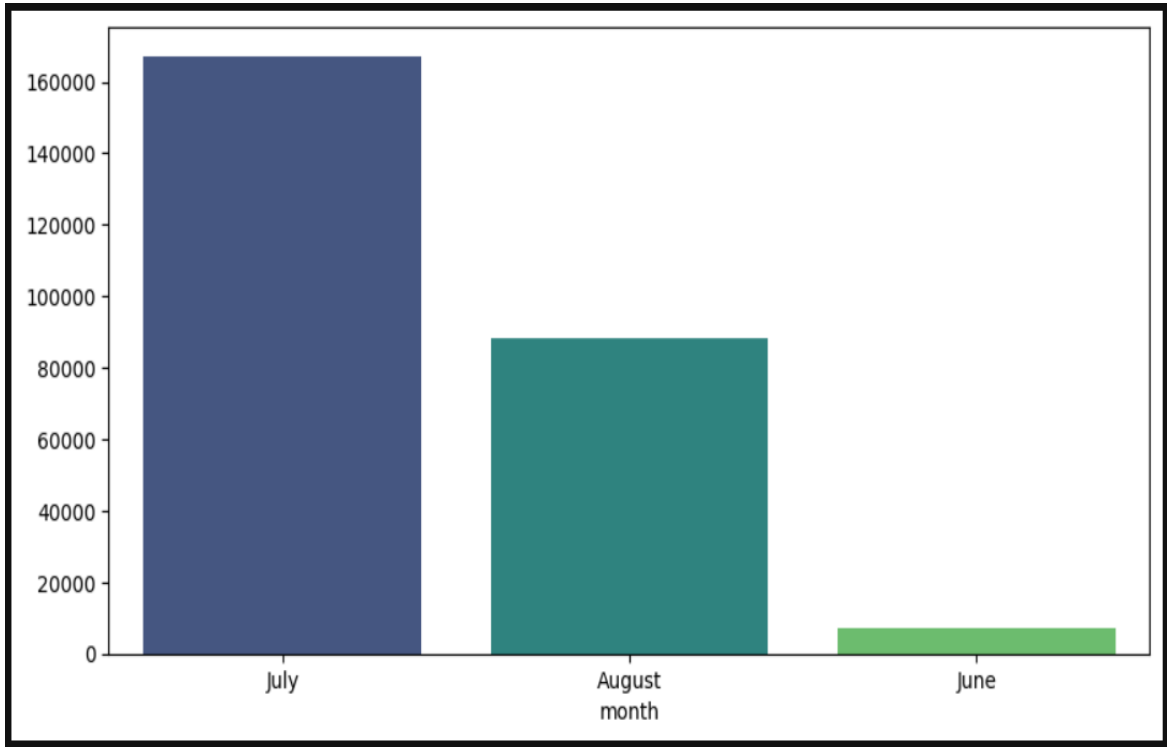
Q. MAXIMUM BOOKINGS WITH RESPECT TO MONTH

```
month
July      167062
August     88423
June        7303
Name: count, dtype: int64
```

Our exploration into booking patterns brought forth an interesting discovery: among the months of June, July, and August, July emerged as the frontrunner, securing the title for the month with the maximum bookings.

Code:

```
booking = pd.read_sql_query("select * from bookings",conn)

booking['book_date'] = pd.to_datetime(booking['book_date'])

booking['month'] = booking['book_date'].dt.month_name()

month_count = booking['month'].value_counts()

print(month_count)


plt.figure(figsize=(10,5))

sns.barplot(x=month_count.index , y=month_count.values,

palette='viridis')

plt.show()
```

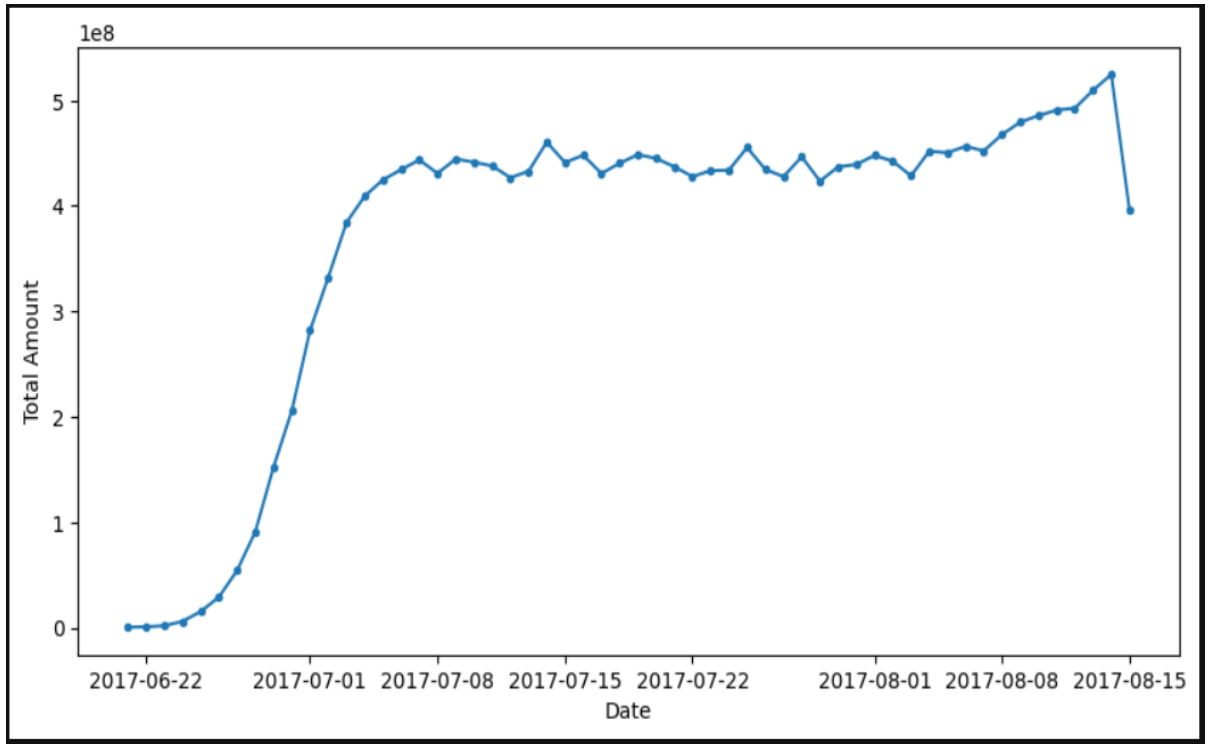


Q. MAXIMUM BOOKINGS PER DAY

Guess what we found while checking when people book the most? Turns out, from August 13th to 15th, loads of folks are making bookings

Code:

```
booking['Date'] = booking['book_date'].dt.date

booking_amount = booking.groupby('Date')[['total_amount']].sum()


plt.figure(figsize=(10,5))

plt.plot(booking_amount.index,

booking_amount['total_amount'], marker = '.')

plt.xlabel("Date")

plt.ylabel("Total Amount")

plt.show()
```



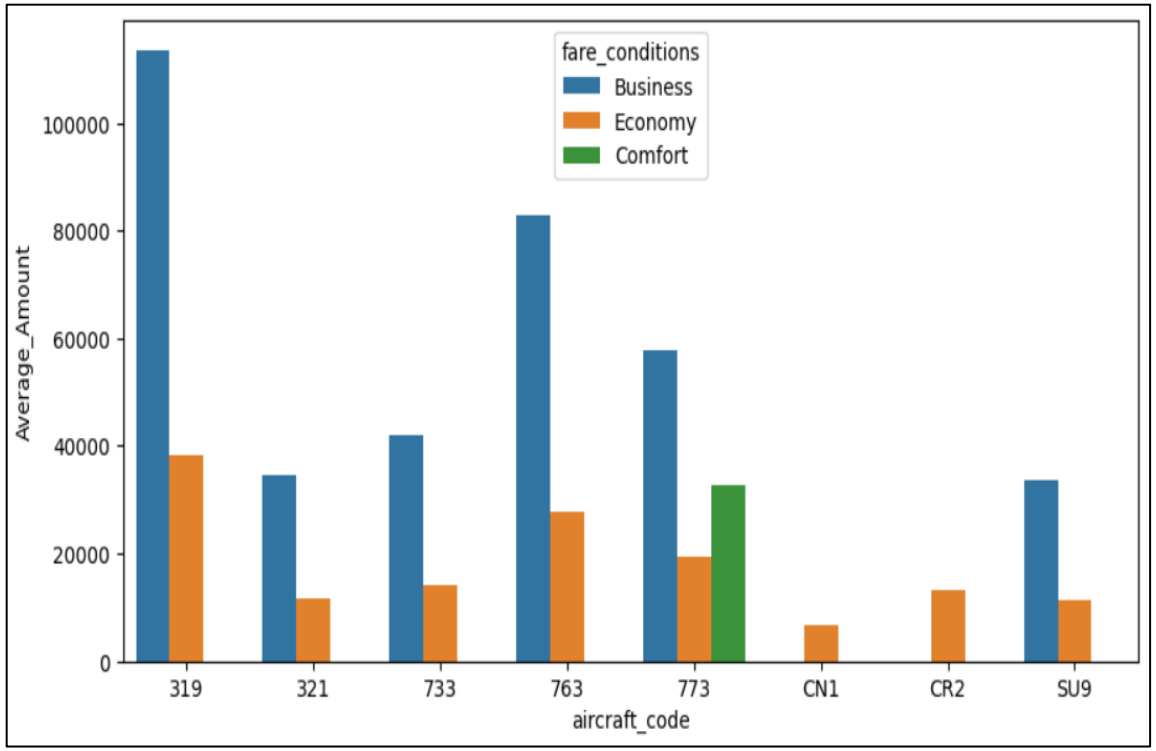
Q. COUNT THE FARE CONDITIONS ON WHICH PASSENGERS USE MOST.

```
fare_conditions
Economy             920793
Business            107642
Comfort              17291
Name: count, dtype: int64
```

In our investigation of passenger fare preferences, the Economy class emerged as the clear favorite, capturing the maximum count. To visualize this, we've crafted a graph that illustrates the distribution of different fare conditions across various aircraft, providing a comprehensive view of passenger choices in each class.

Code:

```
fare_count = pd.read_sql_query("select fare_conditions from ticket_flights", conn)
print(fare_count.value_counts())
df = pd.read_sql_query("SELECT fare_conditions, aircraft_code,
 avg(amount) as Average_Amount from ticket_flights "
                "join flights on ticket_flights.flight_id = flights.flight_id "
                "group by aircraft_code, fare_conditions",conn)

plt.figure(figsize=(10,5))
sns.barplot(data= df, x="aircraft_code",
y="Average_Amount", hue="fare_conditions")
plt.show()
```



Q. FOR EACH AIRCRAFT, CALCULATE THE TOTAL REVENUE AND AVERAGE REVENUE PER TICKET.

We checked how well each plane is doing financially. Some planes are real champs, bringing in lots of money. We looked at total earnings and how much they make on average per ticket. These results give us a peek into which planes are the real money-makers!

Code:

```
average_revenue_per_ticket = pd.read_sql_query("select aircraft_code,Ticket_Count, Total_Revenue, Total_Revenue/Ticket_Count as Average_Revenue from (select aircraft_code, count(*) as Ticket_Count, sum(amount) as Total_Revenue from ticket_flights join flights on ticket_flights.flight_id = flights.flight_id group by aircraft_code)“,conn)

print(average_revenue_per_ticket)
```

	aircraft_code	Ticket_Count	Total_Revenue	Average_Revenue
0	319	52853	2706163100	51201
1	321	107129	1638164100	15291
2	733	86102	1426552100	16568
3	763	124774	4371277100	35033
4	773	144376	3431205500	23765
5	CN1	14672	96373800	6568
6	CR2	150122	1982760500	13207
7	SU9	365698	5114484700	13985

OVERVIEW

CONCLUSION

After a thorough analysis of our airline dataset, several key insights have emerged:

Booking Trends:

•Bookings peak significantly during July, with a notable surge from August 13th to 15th.

Occupancy & Pricing:

•Certain aircraft models, particularly those with over 100 seats, play a crucial role in shaping our fleet strategy.

•Economy class is the popular choice among passengers.

•Pricing strategies should be further optimized, considering factors like occupancy rates and external influences.

Operational Efficiency:

•The financial analysis showcases high-performing aircraft, emphasizing their significant contribution to total and average revenue.

Marketing Impact:

•Past marketing campaigns have impacted bookings positively. Exploring targeted promotions has proven effective.

These findings offer a roadmap for enhancing profitability, improving customer satisfaction, and ensuring optimal operational performance.

FINDINGS

BOOKING TRENDS:

•Bookings peak in July, with a notable surge from August 13th to 15th.

ROUTE OPTIMIZATION:

•Some routes show lower demand, suggesting opportunities for consolidation or adjustments.

AIRCRAFT CAPACITY:

•Aircraft models with over 100 seats play a significant role in the fleet.

•The top 5 planes with more than 100 seats are identified.

MARKETING IMPACT:

•Previous marketing campaigns positively impacted bookings.

•Targeted promotions have been effective.

PRICING AND OCCUPANCY:

•Economy class is the preferred choice among passengers.

•Pricing strategies need optimization, considering occupancy rates and external factors.

OPERATIONAL EFFICIENCY:

•Financial analysis identifies high-performing aircraft contributing significantly to total and average revenue.

SOLUTION

ADDRESSING CN1 AIRCRAFT CANCELLATIONS:

Investigate the root causes of high cancellations associated with the CN1 aircraft. If feasible, implement corrective measures such as maintenance improvements or consider reevaluating the usage of this aircraft in the fleet. If the issues persist, consider temporary removal or replacement with a more reliable aircraft.

INCREASING MONTHLY BOOKINGS WITH OFFERINGS:

Extend marketing efforts and promotional offerings beyond the peak months of July and August. Implement targeted campaigns, discounts, or exclusive deals during other months to stimulate bookings and maintain a consistent revenue flow throughout the year.

EXPANDING COMFORT FARE CONDITION:

Make the "comfort" fare condition available for other aircraft models, not just limited to specific ones. This can attract a broader range of passengers who seek enhanced comfort, potentially leading to increased bookings and revenue. Ensure that the comfort offerings align with customer preferences.

THANK YOU