In [ ]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

ATLIQ HOTELS REVENUE ANALYSIS AND PREDICTION IN INDIA

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns',200)
```

importing data

In [ ]:
```
df_booking=pd.read_csv('/kaggle/input/revenue-insights-hospitality/Revenue_insights_HospitalityDomain/fact_bookings.csv')
df_booking.head(3)
```

In [ ]:
```
df_dim_date=pd.read_csv('/kaggle/input/revenue-insights-hospitality/Revenue_insights_HospitalityDomain/dim_date.csv')
df_dim_date.head(3)
```

In [ ]:
```
df_dim_hotels=pd.read_csv('/kaggle/input/revenue-insights-hospitality/Revenue_insights_HospitalityDomain/dim_hotels.csv')
df_dim_hotels.head(3)
```

In [ ]:
```
df_dim_rooms=pd.read_csv('/kaggle/input/revenue-insights-hospitality/Revenue_insights_HospitalityDomain/dim_rooms.csv')
df_dim_rooms=df_dim_rooms.rename(columns={'room_id':'room_category'})
df_dim_rooms.head(3)
```

convert date column to datetime

In [ ]:
```
# Convert the date column to datetime type
df_dim_date['date'] = pd.to_datetime(df_dim_date['date'])

# Convert the date column to the desired format
df_dim_date['date'] = df_dim_date['date'].dt.strftime('%Y-%m-%d')
df_dim_date=df_dim_date.drop('mmm yy',axis=1)
df_dim_date=df_dim_date.rename(columns={'date':'check_in_date'})
df_dim_date
```

merging all datas for all hotels for analysis

In [ ]:
```
data=df_booking.merge(df_dim_hotels,on='property_id')
data=data.merge(df_dim_rooms,on='room_category')
data=data.merge(df_dim_date,on='check_in_date')

data
```

data preprocessing

In [ ]:
```
data.isna().sum()
```

droping unnecessary features

In [ ]:
```
data=data.drop(['ratings_given','booking_id','booking_date'],axis=1)
data
```

In [ ]:
```
data.info()
```

In [ ]:
```
data['check_in_date'] = pd.to_datetime(data['check_in_date'])
data['checkout_date'] = pd.to_datetime(data['checkout_date'])
```

In [ ]:
```
data.info()
```

In [ ]:
```
data.shape
```

In [ ]:
```
data.describe()
```

extract day name and month feature

In [ ]:
```
# Extract month from check_in_date
data['month'] = data['check_in_date'].dt.month

# Extract day from check_in_date
data['day'] = data['check_in_date'].dt.day

# Extract day name (e.g., Monday, Tuesday) from check_in_date
data['day_name'] = data['check_in_date'].dt.day_name()
```

In [ ]:
```
data
```

extract week number from week feature and removing every thing else to analysis

In [ ]:
```
import re

# Extract number from "W 32" format
data['week no'] = data['week no'].apply(lambda x: re.findall(r'\d+', x)[0] if isinstance(x, str) else x)
data['week no']=pd.to_numeric(data['week no'])
data
```

In [ ]:
```
data.to_csv('data.csv',index=False)
```

In [ ]:
```
data.info()
```

checking cancelled booking percentage

In [ ]:
```
data['booking_status'].value_counts(normalize='True')
```

In [ ]:
```
data['city'].value_counts().plot(kind='bar')
```

Mumbai city is the most booking city for Atliq Hotels

In [ ]:
```
data.head()
```

In [ ]:
```
data['day_type'].value_counts().plot(kind='pie',autopct='%1.1f%%')
```

In [ ]:
```
data.groupby('property_name')['revenue_realized'].sum()
```

In [ ]:
```
import plotly.express as px

# Group the data by 'property_name' and calculate the sum of 'revenue_realized'
grouped_data = data.groupby('property_name')['revenue_realized'].sum().reset_index().sort_values(by='revenue_realized',ascending=False)

# Plot the grouped data using Plotly Express
fig = px.bar(grouped_data, x='property_name', y='revenue_realized', title='Total Revenue by Property')
fig.show()
```

Atliq Exotica is the highest revenue hotel and Atliq Seasons is the lowest revenue hotel

there is a problem in Atliq Seasons make the bookings are very low so the revenue is very low

In [ ]:
```
data['booking_status'].value_counts().plot(kind='pie',autopct='%1.1f%%')
```

the cancelled bookings are about 25% from all bookings

In [ ]:
```
data['booking_platform'].value_counts().plot(kind='pie',autopct='%1.1f%%')
```

the highest bookings platform is makeyourtrip platform and the direct offline booking is the lowest

In [ ]:
```
grouped_data = data.groupby('booking_platform')['revenue_realized'].sum().reset_index().sort_values(by='revenue_realized',ascending=False)
grouped_data
```

In [ ]:
```
# Group the data by 'property_name' and calculate the sum of 'revenue_realized'
grouped_data = data.groupby('booking_platform')['revenue_realized'].sum().reset_index().sort_values(by='revenue_realized',ascending=False)

# Plot the grouped data using Plotly Express
fig = px.bar(grouped_data, x='booking_platform', y='revenue_realized', title='Total Revenue by Platform')
fig.show()
```

In [ ]:
```
grouped_data = data.groupby('city')['revenue_realized'].sum().reset_index().sort_values(by='revenue_realized',ascending=False)
grouped_data
```

In [ ]:
```
# Group the data by 'property_name' and calculate the sum of 'revenue_realized'
grouped_data = data.groupby('city')['revenue_realized'].sum().reset_index().sort_values(by='revenue_realized',ascending=False)
# Plot the grouped data using Plotly Express
fig = px.bar(grouped_data, x='city', y='revenue_realized', title='Total Revenue by city')
fig.show()
```

the highest revenue city is mumbai and the lowest is delhi

In [ ]:
```
data.head()
```

drop unnecessary features to predict the revenue

In [ ]:
```
data=data.drop(['property_id','check_in_date','checkout_date','week no'],axis=1)
```

In [ ]:
```
data.head()
```

convert categorical features to numerical by label encoder

In [ ]:
```
from sklearn.preprocessing import LabelEncoder

# Create an instance of LabelEncoder
label_encoder = LabelEncoder()

# Loop over each column in the DataFrame
for column in data.columns:
    # Check if the column is categorical (dtype 'object')
    if data[column].dtype == 'object':
        # Apply label encoding to the column
        data[column] = label_encoder.fit_transform(data[column])
```

In [ ]:
```
data
```

In [ ]:
```
X=data.drop('revenue_realized',axis=1)
y=data.revenue_realized
```

split the data to train and test

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.25,random_state=42)
```

In [ ]:
```
X_train.shape
```

In [ ]:
```
X_test.shape
```

using XGBOOST REGRESSOR for predicting

In [ ]:
```
import xgboost as xgb
from sklearn.metrics import mean_absolute_error
# Instantiate the XGBoost regressor
model = xgb.XGBRegressor(n_estimators=100,learning_rate=.05)

# Fit the model on the training data
model.fit(X_train, y_train)

# Predict using the trained model
prediction = model.predict(X_test)
print(mean_absolute_error(y_test, prediction))
```

In [ ]:
```
prediction[:10]
```

In [ ]:
```
y_test
```

comparing between actual and prediction data

In [ ]:
```
pred_df=pd.DataFrame(prediction,index=X_test.index,columns=['prediction'])
# Merge the predicted values with the actual test data
merged_df = pd.concat([y_test,pred_df], axis=1)
merged_df
```

as we see the mean absolute error for xgboost regressor is very low and the difference between actual and prediction is very low so we will use xgb for prediction