DAV 5400 Module 6 Project

by Sreyash Mudiam Venkata

Introduction

- For this research project, I have chosen to work with a dataset that contains information related to domestic flight routes within the United States from 1990 to 2009.
- I picked this dataset as anything related to flights and its operations is of huge interest to me as I am a former Aerospace Engineer.
- The dataset includes details such as the origin and destination of flights, the number of passengers, available seats, flight frequencies, distances, flight dates, and population figures of the origin and destination cities.
- This research might be able to provide valuable insights into an important aspect of the aviation industry and contribute to a better understanding of factors affecting seat occupancy, distance, frequency of domestic flights.

Research Question

- The primary research question that I have chosen in this project is to understand the factors influencing seat occupancy rates, busiest flight routes based on cities and states for the domestic flights within the United States. To explore the relationship between various attributes, such as distance, flight frequency, and passenger numbers, to gain insights into the determinants of seat occupancy in these flights.
- Through a combination of exploratory data analysis and inference, I will uncover key findings that shed light on the dynamics of seat occupancy in domestic flights. My analysis includes the use of both Matplotlib and Seaborn for visualizations, ensuring a comprehensive exploration and valid conclusions from the dataset.

Data Summary

- The dataset consists of records for numerous domestic flight routes within the United States from year 1990 to 2009 from which various number of use cases can be derived as shown further.
- Attributes for each use case includes:
  - Origin: The departure airport code.
  - Origin City: The departure city with State Code.
  - Destination: The arrival airport code.
  - Destination City: The arrival city with State Code.
  - Passengers: The number of passengers on the flight.
  - Seats: The number of available seats on the flight.
  - Flights: The number of flights for that specific route.
  - Distance: The distance of the flight route in miles.
  - Fly Date: The date of the flight in yyyymm format.
  - Origin Population: The population of the origin city.
  - Destination Population: The population of the destination city.

- Importing the M6Project class from from M6_Project python file

In [1]:
```
from M6_Project import M6Project
```

- Initializing the class
- Providing path of the data file

In [2]:
```
tasks = M6Project('/Users/sreyashvenkata/Downloads/Katz DAV/Analytics Progra
```

```
        Origin Destination        Origin City           Destination City  \
0          MHK         AMW       Manhattan, KS                  Ames, IA
1          EUG         RDM          Eugene, OR                  Bend, OR
2          EUG         RDM          Eugene, OR                  Bend, OR
3          EUG         RDM          Eugene, OR                  Bend, OR
4          MFR         RDM         Medford, OR                  Bend, OR
...        ...         ...                 ...                       ...
3606798    STL         TBN       St. Louis, MO      Fort Leonard Wood, MO
3606799    STL         TBN       St. Louis, MO      Fort Leonard Wood, MO
3606800    STL         TBN       St. Louis, MO      Fort Leonard Wood, MO
3606801    CGI         TBN  Cape Girardeau, MO      Fort Leonard Wood, MO
3606802    FWA         OH1      Fort Wayne, IN  Washington Court House, OH

         Passengers  Seats  Flights  Distance  Fly Date  Origin Population  \
0                21     30        1     254.0    200810             122049
1                41    396       22     103.0    199011             284093
2                88    342       19     103.0    199012             284093
3                11     72        4     103.0    199010             284093
4                 0     18        1     156.0    199002             147300
...             ...    ...      ...       ...       ...                ...
3606798         281    969       51     119.0    200902            2828990
3606799         245   1026       54     119.0    200911            2828990
3606800         363   1273       67     119.0    200908            2828990
3606801           2     19        1     146.0    200908              93712
3606802           0      0        1     135.0    200309             398574

         Destination Population
0                         86219
1                         76034
2                         76034
3                         76034
4                         76034
...                         ...
3606798                   46457
3606799                   46457
3606800                   46457
3606801                   46457
3606802                   28133

[3606803 rows x 11 columns]
```

Data Wrangling

- For deriving better conclusions and good understanding, it is better to split Origin City, Destination City and Fly Date(yyyymm) to Origin_City and Origin_State, Destination_City and Destination_State, Year(yyyy) and Month(mm) respectively.
- This can be achieved by:
  - Splitting the 'Origin City' and 'Destination City' columns into 'Origin_City' and 'Origin_State' columns and 'Destination_City' and 'Destination_State' columns
  - Drop the original 'Origin City' and 'Destination City' columns.
  - Convert the 'Fly Date' column to a string.
  - Extract year and month using string slicing.
  - Convert the new columns to integers.
  - drop the 'Fly Date' column

In [3]:
```
tasks.data_wrangling()
```

```
  Origin Destination  Passengers  Seats  Flights  Distance  Origin Populatio
n  \
0    MHK         AMW          21     30        1     254.0            12204
9
1    EUG         RDM          41    396       22     103.0            28409
3
2    EUG         RDM          88    342       19     103.0            28409
3
3    EUG         RDM          11     72        4     103.0            28409
3
4    MFR         RDM           0     18        1     156.0            14730
0

   Destination Population Origin_City Origin_State Destination_City  \
0                   86219   Manhattan           KS             Ames
1                   76034      Eugene           OR             Bend
2                   76034      Eugene           OR             Bend
3                   76034      Eugene           OR             Bend
4                   76034     Medford           OR             Bend

  Destination_State  Year  Month
0                IA  2008     10
1                OR  1990     11
2                OR  1990     12
3                OR  1990     10
4                OR  1990      2
```

Desired Columns Order

- Arranging the columns in a desired order is necessary for using and reading it efficiently. All the origin details are moved to front followed by destination details and then remaining numeric columns. This can be achieved by:
  - Defining the desired column order
  - Reordering the DataFrame columns

In [4]:
```
tasks.desired_order()
```

```
  Origin Origin_City Origin_State Destination Destination_City  \
0    MHK   Manhattan           KS         AMW             Ames   
1    EUG      Eugene           OR         RDM             Bend   
2    EUG      Eugene           OR         RDM             Bend   
3    EUG      Eugene           OR         RDM             Bend   
4    MFR     Medford           OR         RDM             Bend   

  Destination_State  Year  Month  Passengers  Seats  Flights  Distance  \
0                IA  2008     10          21     30        1     254.0   
1                OR  1990     11          41    396       22     103.0   
2                OR  1990     12          88    342       19     103.0   
3                OR  1990     10          11     72        4     103.0   
4                OR  1990      2           0     18        1     156.0   

   Origin Population  Destination Population  
0             122049                   86219  
1             284093                   76034  
2             284093                   76034  
3             284093                   76034  
4             147300                   76034  
```

Data Shape and Type

- Let's get the number of rows and columns in the dataset and then list the column names and their data types to know about the data shape and data types
- Typically
  - Origin and Destination are stored as strings (object data type).
  - Passengers, Seats, Flights, and Distance are usually stored as integers or floating-point numbers.
  - Fly Date may be stored as a date or datetime data type, here we converted them into int64.
  - Origin Population and Destination Population are also stored as integers or floating-point numbers.

In [5]:
```
tasks.shape()
```

```
Number of Rows: 3606803
Number of Columns: 14
Column Data Types:
Origin                       object
Origin_City                  object
Origin_State                 object
Destination                  object
Destination_City             object
Destination_State            object
Year                          int64
Month                         int64
Passengers                    int64
Seats                         int64
Flights                       int64
Distance                    float64
Origin Population             int64
Destination Population        int64
dtype: object
```

Exploratory Data Analysis (EDA)

- It is a critical step in the data analysis process as it involves investigating, summarizing, and visualizing the main characteristics of a dataset to better understand its structure, detect patterns, identify anomalies, and extract insights.
- Summary Statistics: Let's calculate and examine basic statistics for each attribute to understand the central tendency, dispersion, and other key properties of the data. It commonly include mean, median, standard deviation, minimum, maximum, and quartiles, it can be derived easily by using describe function as follows.
- Let us set the float format for display instead of scientific notation.

In [6]:
```
tasks.summary()
```

```
             Year      Month   Passengers      Seats    Flights   Distance  \
count 3606803.0  3606803.0    3606803.0  3606803.0  3606803.0  3606803.0
mean     2000.6        6.5       2688.9     4048.3       37.2      697.3
std         5.7        3.5       4347.6     6200.9       49.6      604.4
min      1990.0        1.0          0.0        0.0        0.0        0.0
25%      1996.0        4.0        109.0      156.0        2.0      273.0
50%      2001.0        7.0       1118.0     1998.0       25.0      519.0
75%      2006.0       10.0       3503.0     5370.0       55.0      927.0
max      2009.0       12.0      89597.0   147062.0     1128.0     5095.0

       Origin Population  Destination Population
count          3606803.0               3606803.0
mean           5871502.5               5897982.4
std            7858061.6               7906127.4
min              13005.0                 12887.0
25%            1030597.0               1025470.0
50%            2400193.0               2400193.0
75%            8613622.0               8635706.0
max           38139592.0              38139592.0
```

- Let's identify unique values in the categorical columns which are non-numeric.

In [7]:
```
tasks.unique_values()
```

```
Unique values in 'Origin': ['MHK' 'EUG' 'MFR' 'SEA' 'PDX' 'LMT' 'SFO' 'LAX'
'EAT' 'YKM' 'EKO' 'SLE'
 'GEG' 'RDD' 'LWS' 'AST' 'CLM' 'PDT' 'SJC' 'ACV' 'PUW' 'SMF' 'FLL' 'PHX'
 'BFI' 'GGG' 'GTF' 'FAT' 'TUS' 'MWH' 'BIL' 'DFW' 'RBG' 'SLC' 'CPR' 'RDM'
 'RNO' 'FBK' 'ANC' 'BIF' 'CIC' 'SAF' 'ABQ' 'FSM' 'LAS' 'DQF' 'ICT' 'AZA'
 'DLH' 'CEC' 'OKC' 'SAN' 'STS' 'DRO' 'OAK' 'IAH' 'ELP' 'DBQ' 'HLN' 'TUL'
 'OMA' 'ACT' 'BTM' 'AUS' 'PIA' 'CWA' 'PUB' 'FOE' 'COS' 'GJT' 'LBF' 'SPS'
 'CYS' 'ABR' 'BRD' 'AMA' 'ROW' 'BIS' 'MSO' 'GCC' 'LBB' 'LBL' 'MAF' 'MSN'
 'LNK' 'ALO' 'RFD' 'GRB' 'LSE' 'FCA' 'MKE' 'BLI' 'FOD' 'DSM' 'EAU' 'MCW'
```

```
 'RAP' 'LIT' 'SHV' 'MCI' 'GFK' 'BFL' 'SAT' 'MSP' 'SGF' 'BFF' 'FSD' 'GRI'
 'CID' 'FYV' 'SUX' 'GCK' 'BTR' 'IDA' 'SPI' 'SBA' 'CRP' 'DEC' 'ORD' 'BZN'
 'PIR' 'MLU' 'YUM' 'MOT' 'GYY' 'FAR' 'RST' 'STL' 'BNA' 'EVV' 'MEM' 'JAN'
 'MOD' 'CMI' 'SCK' 'IND' 'BMI' 'TWF' 'LRD' 'CAK' 'MTJ' 'CLE' 'CLL' 'DTW'
 'ABI' 'DAY' 'CVG' 'JAC' 'CMH' 'SBN' 'FWA' 'AZO' 'HOU' 'RKS' 'BPT' 'PIH'
 'SLN' 'UIN' 'LAW' 'TOL' 'RIW' 'BRO' 'SJT' 'PRB' 'SHR' 'MCE' 'GRK' 'VGT'
 'CDC' 'PNC' 'HRO' 'WDG' 'RCA' 'BWD' 'DDC' 'PIT' 'ORH' 'ITH' 'ELM' 'BOS'
 'ATL' 'MDW' 'PHL' 'GSO' 'FNT' 'BUF' 'EWR' 'BGR' 'CLT' 'JFK' 'SYR' 'RIC'
 'GRR' 'YIP' 'ABE' 'LGA' 'PVD' 'LEX' 'BHM' 'ACY' 'MCO' 'YNG' 'PWM' 'MDT'
 'BGM' 'IAD' 'ALB' 'MSY' 'ROC' 'AVP' 'ROA' 'RDU' 'GSP' 'BTV' 'CRW' 'CHS'
 'DCA' 'CHA' 'SCE' 'CHO' 'LAN' 'BDL' 'CAE' 'JHW' 'MHT' 'ILM' 'GDC' 'FLO'
 'ISO' 'BWI' 'ERI' 'SFB' 'PGV' 'TYS' 'HSV' 'BWG' 'ADS' 'DET' 'LOZ' 'SUS'
 'WGO' 'GSB' 'MYR' 'MBS' 'CLU' 'LCK' 'SYI' 'TLH' 'BMG' 'MNN' 'LUK' 'MGM'
 'MFD' 'PNS' 'ASN' 'TVC' 'KY5' 'MIA' 'GPT' 'TCL' 'DRT' 'AEX' 'HUF' 'RWI'
 'TPA' 'SNA' 'AGC' 'GBD' 'FAI' 'EAR' 'BRL' 'JLN' 'TBN' 'HNL' 'OGG' 'NFL'
 'DAL' 'MHR' 'LFT' 'SCF' 'HIK' 'KTN' 'LCH' 'ILN' 'MOB' 'FWH' 'LGU' 'XNA'
 'LRU' 'RIV' 'COU' 'ILE' 'ESF' 'TYR' 'JAX' 'PBI' 'SAV' 'FLG' 'EFD' 'AFW'
 'CNW' 'SWO' 'SKF' 'CSG' 'ELD' 'FFO' 'PAM' 'GNV' 'HMN' 'ILG' 'DMA' 'FTW'
 'HII' 'IPL' 'POB' 'TCM' 'SBP' 'TTN' 'ATW' 'BTL' 'PDK' 'LAF' 'RDG' 'MKG'
 'HGR' 'MQT' 'NZC' 'UCA' 'FAY' 'AGS' 'AVL' 'VLD' 'EKA' 'DOV' 'NKX' 'LUF'
 'CVS' 'BAD' 'LSV' 'TIK' 'EIL' 'JMS' 'MWA' 'GLH' 'EKI' 'VWL' 'SZL' 'LAR'
 'OSH' 'MOR' 'PWK' 'APN' 'IRK' 'PAH' 'MRC' 'CSV' 'GMU' 'DPA' 'DCU' 'SKY'
 'ADM' 'CNM' 'CVN' 'HOB' 'FMN' 'RUI' 'BKX' 'HON' 'BQK' 'ABY' 'DHN' 'OAJ'
 'MEI' 'GTR' 'PFN' 'LYH' 'MCN' 'EYW' 'BKL' 'ARA' 'APF' 'EWN' 'AHN' 'EDF'
 'DYS' 'BYH' 'MCF' 'MIQ' 'NQX' 'LTS' 'TNT' 'OPF' 'HKY' 'CKB' 'NPA' 'WRB'
 'TUP' 'PIB' 'NJK' 'LSF' 'OCF' 'MIE' 'NEW' 'SVN' 'ORL' 'NIP' 'FXE' 'NBG'
 'SDM' 'MXF' 'SSC' 'JST' 'HKS' 'AND' 'BFM' 'TMB' 'HIF' 'HOT' 'PNE' 'MTN'
 'LAL' 'MKC' 'SHD' 'XXW' 'MIO' 'MIB' 'ISN' 'DIK' 'BMC' 'SGU' 'MUO' 'OGD'
 'SAW' 'OFK' 'NZY' 'CVO' 'ITO' 'MDH' 'AID' 'SGH' 'OLS' 'ANB' 'FDY' 'BDR'
 'HVN' 'RUT' 'OGS' 'RKD' 'ART' 'STC' 'AUO' 'PBG' 'IMT' 'PVU' 'CGI' 'PKB'
 'THV' 'AUG' 'SBY' 'CUB' 'PGD' 'FVS' 'SVC' 'CRS' 'IPT' 'JBR' 'SQI' 'EGP'
 'ATY' 'GAD' 'VCT' 'RMG' 'MCC' 'STJ' 'BKG' 'BJI' 'CHI' 'MKL' 'MVN' 'HLM'
 'MGY' 'LEW' 'BFD' 'GUP' 'UKI' 'OXR' 'VIS' 'SOW' 'HVR' 'HYS' 'WFB' 'JNU'
 'JSE' 'ADQ' 'KDK' 'AOO' 'MS1' 'NQA' 'HUM' 'MML' 'MMI' 'IRS' 'JEF' 'MPB'
 'SRC' 'GLW' 'SPA' 'TN6' 'CEV' 'SER' 'BGD' 'AIY' 'JRA' 'JRB' 'TSS' 'LNS'
 'SUM' 'ALW' 'SAC' 'OKK' 'POU' 'VAD' 'LKE' 'SKA' 'BSM' 'GRF' 'DLF' 'APC'
 'ESC' 'CWI' 'BGS' 'SWW' 'BFR' 'ZZV' 'ISM' 'WQM' 'AXN' 'FFM' 'GWO' 'FEP'
 'KY1' 'GVL' 'AOH' 'STP' 'BKW' 'DNN' 'CPS' 'PSF' 'DUC' 'UVA' 'MOP' 'BJJ'
 'JXN' 'OWB' 'SIK' 'AXV' 'DMO' 'FFT' 'II2' 'DNV' 'SAD' 'FTY' 'UCY' 'PHT'
 'IN1' 'TDZ' 'NC3' 'EKX' 'TN3' 'PBF' 'BVX' 'THA' 'OH5' 'SBM' 'KY3' 'OH3'
 'ATO' 'VWT' 'WV1' 'DVN' 'CGF' 'OH2' 'SRW' 'YKN' 'OTM' 'GCY' 'GUS' 'MMT'
 'LUL' 'OFF' 'ALM' 'PUC' 'LFK' 'BBC' 'VWH' 'SEM' 'RID' 'SVH' 'MI2' 'VEL'
 'MKT' 'ASL' 'HFD' 'PRC' 'DVT' 'AR1' 'IAB' 'ODW' 'OLM' 'END' 'MVW' 'GBG'
 'PWT' 'SHN' 'ELN' 'LGD' 'SFF' 'VWD' 'MAE' 'HUT' 'SSI' 'UBS' 'CBM' 'GFL'
 'NY3' '1B1' 'FCH' 'WTC' 'PWA' 'TIW' 'PMH' 'CAD' 'MGW' 'TBR' 'CBE' 'MDD'
 'MRI' 'LSD' 'HCA' 'SLB' 'DQU' 'OSU' 'MYF' 'SEE' 'AMW' 'LWC' 'MZZ' 'OWA'
 'DNE' 'HSH' 'LWF' 'HBG' 'GRD' 'JZU' 'AMK' 'RND' 'IKK' 'FET' 'TMA' 'ECG'
 'STF' 'LAM' 'TSM' 'TX6' 'OH1' 'S27' 'DQC' 'MHL' 'MHE' 'TKF' 'CGX' 'WMH'
 'LXN' 'HSI' 'WBR' 'GGE' 'HLG' 'LHV' 'CRE' 'BOK' 'BIH' 'MQJ' 'LCI']
Unique values in 'Origin_City': ['Manhattan' 'Eugene' 'Medford' 'Seattle' 'P
ortland' 'Klamath Falls'
 'San Francisco' 'Los Angeles' 'Wenatchee' 'Yakima' 'Elko' 'Salem'
 'Spokane' 'Redding' 'Lewiston' 'Astoria' 'Port Angeles' 'Pendleton'
```

```
'San Jose' 'Eureka' 'Pullman' 'Sacramento' 'Fort Lauderdale' 'Phoenix'
'Longview' 'Great Falls' 'Fresno' 'Tucson' 'Moses Lake' 'Billings'
'Dallas' 'Roseburg' 'Salt Lake City' 'Casper' 'Bend' 'Reno' 'Fairbanks'
'Anchorage' 'El Paso' 'Chico' 'Santa Fe' 'Albuquerque' 'Fort Smith'
'Las Vegas' 'Wichita' 'Duluth' 'Crescent City' 'Oklahoma City'
'San Diego' 'Santa Rosa' 'Durango' 'Oakland' 'Houston' 'Dubuque' 'Helena'
'Tulsa' 'Omaha' 'Waco' 'Butte' 'Austin' 'Peoria' 'Wausau' 'Pueblo'
'Topeka' 'Colorado Springs' 'Grand Junction' 'North Platte'
'Wichita Falls' 'Cheyenne' 'Aberdeen' 'Brainerd' 'Amarillo' 'Roswell'
'Bismarck' 'Missoula' 'Gillette' 'Lubbock' 'Liberal' 'Midland' 'Madison'
'Lincoln' 'Waterloo' 'Rockford' 'Green Bay' 'La Crosse' 'Kalispell'
'Milwaukee' 'Bellingham' 'Fort Dodge' 'Des Moines' 'Eau Claire'
'Mason City' 'Rapid City' 'Little Rock' 'Shreveport' 'Kansas City'
'Grand Forks' 'Bakersfield' 'San Antonio' 'Minneapolis' 'Springfield'
'Scottsbluff' 'Sioux Falls' 'Grand Island' 'Cedar Rapids' 'Fayetteville'
'Sioux City' 'Garden City' 'Baton Rouge' 'Idaho Falls' 'Santa Barbara'
'Corpus Christi' 'Decatur' 'Chicago' 'Bozeman' 'Pierre' 'Monroe' 'Yuma'
'Minot' 'Gary' 'Fargo' 'Rochester' 'St. Louis' 'Nashville' 'Evansville'
'Memphis' 'Jackson' 'Modesto' 'Champaign' 'Stockton' 'Indianapolis'
'Bloomington' 'Twin Falls' 'Laredo' 'Akron' 'Montrose' 'Cleveland'
'College Station' 'Detroit' 'Abilene' 'Dayton' 'Cincinnati' 'Columbus'
'South Bend' 'Fort Wayne' 'Kalamazoo' 'Rock Springs' 'Beaumont'
'Pocatello' 'Salina' 'Quincy' 'Lawton' 'Toledo' 'Riverton' 'Brownsville'
'San Angelo' 'San Luis Obispo' 'Sheridan' 'Merced' 'Killeen' 'Cedar City'
'Ponca City' 'Harrison' 'Enid' 'Brownwood' 'Dodge City' 'Pittsburgh'
'Worcester' 'Ithaca' 'Elmira' 'Boston' 'Atlanta' 'Philadelphia'
'Greensboro' 'Flint' 'Buffalo' 'Newark' 'Bangor' 'Charlotte' 'New York'
'Syracuse' 'Richmond' 'Grand Rapids' 'Allentown' 'Providence' 'Lexington'
'Birmingham' 'Atlantic City' 'Orlando' 'Youngstown' 'Harrisburg'
'Binghamton' 'Washington' 'Albany' 'New Orleans' 'Scranton' 'Roanoke'
'Raleigh' 'Greenville' 'Burlington' 'Charleston' 'Chattanooga'
'State College' 'Charlottesville' 'Lansing' 'Hartford' 'Columbia'
'Jamestown' 'Manchester' 'Wilmington' 'Florence' 'Kinston' 'Baltimore'
'Erie' 'Knoxville' 'Huntsville' 'Bowling Green' 'London' 'Winchester'
'Goldsboro' 'Myrtle Beach' 'Saginaw' 'Shelbyville' 'Tallahassee' 'Marion'
'Montgomery' 'Mansfield' 'Pensacola' 'Talladega' 'Traverse City'
'Madisonville' 'Miami' 'Gulfport' 'Tuscaloosa' 'Del Rio' 'Alexandria'
'Terre Haute' 'Rocky Mount' 'Tampa' 'Santa Ana' 'Great Bend' 'Kearney'
'Joplin' 'Fort Leonard Wood' 'Honolulu' 'Kahului' 'Fallon' 'Lafayette'
'Ketchikan' 'Lake Charles' 'Mobile' 'Logan' 'Las Cruces' 'Riverside'
'Tyler' 'Jacksonville' 'West Palm Beach' 'Savannah' 'Flagstaff'
'Stillwater' 'El Dorado' 'Panama City' 'Gainesville' 'Alamogordo'
'Lake Havasu City' 'El Centro' 'Tacoma' 'Trenton' 'Appleton'
'Battle Creek' 'Reading' 'Muskegon' 'Hagerstown' 'Marquette' 'Utica'
'Augusta' 'Asheville' 'Valdosta' 'Dover' 'Clovis' 'Elkhart' 'Bemidji'
'Warrensburg' 'Laramie' 'Oshkosh' 'Morristown' 'Alpena' 'Kirksville'
'Paducah' 'Crossville' 'Sandusky' 'Ardmore' 'Carlsbad' 'Hobbs'
'Farmington' 'Ruidoso' 'Brookings' 'Huron' 'Brunswick' 'Dothan'
'Meridian' 'Lynchburg' 'Macon' 'Key West' 'New Iberia' 'Naples'
'New Bern' 'Athens' 'Blytheville' 'Altus' 'Hickory' 'Clarksburg' 'Tupelo'
'Hattiesburg' 'Ocala' 'Muncie' 'Sumter' 'Johnstown' 'Anderson' 'Ogden'
'Hot Springs' 'Lakeland' 'Staunton' 'Lake City' 'Williston' 'Dickinson'
'Brigham City' 'St. George' 'Mountain Home' 'Norfolk' 'Hilo' 'Carbondale'
```

```
 'Nogales' 'Anniston' 'Findlay' 'Bridgeport' 'New Haven' 'Rutland'
 'Ogdensburg' 'Rockland' 'Watertown' 'St. Cloud' 'Auburn' 'Plattsburgh'
 'Iron Mountain' 'Provo' 'Cape Girardeau' 'Parkersburg' 'York' 'Salisbury'
 'Punta Gorda' 'Rexburg' 'Silver City' 'Corsicana' 'Williamsport'
 'Jonesboro' 'Sterling' 'Eagle Pass' 'Gadsden' 'Victoria' 'Rome'
 'St. Joseph' 'Branson' 'Mount Vernon' 'Holland' 'Bradford' 'Gallup'
 'Ukiah' 'Oxnard' 'Visalia' 'Show Low' 'Havre' 'Hays' 'Juneau' 'Kodiak'
 'Altoona' 'Grenada' 'Houma' 'Marshall' 'Sturgis' 'Jefferson City'
 'Searcy' 'Glasgow' 'Spartanburg' 'Dyersburg' 'Connersville' 'Seymour'
 'Borger' 'Lancaster' 'Walla Walla' 'Kokomo' 'Poughkeepsie' 'Napa'
 'Escanaba' 'Clinton' 'Big Spring' 'Sweetwater' 'Bedford' 'Zanesville'
 'Fergus Falls' 'Greenwood' 'Freeport' 'Danville' 'Lima' 'Beckley'
 'Dalton' 'Pittsfield' 'Duncan' 'Uvalde' 'Mount Pleasant' 'Wooster'
 'Owensboro' 'Sikeston' 'Wapakoneta' 'Sedalia' 'Frankfort' 'Safford'
 'Union City' 'Paris' 'Shelby' 'Elizabethtown' 'Lewisburg' 'Pine Bluff'
 'Batesville' 'Tullahoma' 'Sheboygan' 'Mount Sterling' 'Chillicothe'
 'Americus' 'Davenport' 'Bellefontaine' 'Yankton' 'Ottumwa' 'Greeneville'
 'Peru' 'Laurel' 'Price' 'Nacogdoches' 'Bay City' 'Selma' 'Statesville'
 'Vernal' 'Mankato' 'Prescott' 'Russellville' 'Oak Harbor' 'Olympia'
 'Galesburg' 'Bremerton' 'Shelton' 'Ellensburg' 'La Grande' 'Hanford'
 'Madera' 'Hutchinson' 'Glens Falls' 'Oneonta' 'Hudson' 'Portsmouth'
 'Cadillac' 'Morgantown' 'Statesboro' 'Cumberland' 'Storm Lake' 'Ames'
 'Lawrence' 'Owatonna' 'Lawrenceburg' 'Bennington' 'Kankakee' 'Fremont'
 'Tifton' 'Elizabeth City' 'Starkville' 'Los Alamos' 'Taos'
 'Washington Court House' 'Sebastian' 'Mitchell' 'Truckee' 'Hastings'
 'Big Rapids' 'Georgetown' 'Wheeling' 'Lock Haven' 'Bishop' 'Laconia']
Unique values in 'Origin_State': ['KS' 'OR' 'WA' 'CA' 'NV' 'ID' 'FL' 'AZ' 'T
X' 'MT' 'UT' 'WY' 'AK' 'NM'
 'AR' 'MN' 'OK' 'CO' 'IA' 'NE' 'IL' 'WI' 'SD' 'ND' 'LA' 'MO' 'IN' 'TN'
 'MS' 'OH' 'MI' 'PA' 'MA' 'NY' 'GA' 'NC' 'NJ' 'ME' 'VA' 'RI' 'KY' 'AL'
 'DC' 'SC' 'VT' 'WV' 'CT' 'NH' 'MD' 'HI' 'DE']
Unique values in 'Destination': ['AMW' 'RDM' 'EKO' 'WDG' 'END' 'ERI' 'GYY' '
HYS' 'ITO' 'AOH' 'APC' 'GUS'
 'RNO' 'RMG' 'TSM' 'ACT' 'CNW' 'THV' 'YUM' 'CAK' 'LTS' 'BTM' 'CIC' 'DOV'
 'FAR' 'FNT' 'HVR' 'HOB' 'HUM' 'HON' 'LGU' 'MCN' 'WRB' 'MIA' 'TNT' 'OPF'
 'MPB' 'TMB' 'MIO' 'MOT' 'MIB' 'OCF' 'OGD' 'HIF' 'OMA' 'MIQ' 'OFF' 'PHT'
 'PUC' 'PVU' 'SLE' 'SEM' 'TPA' 'MCF' 'TUL' 'RVS' 'TYR' 'UKI' 'UCA' 'ABY'
 'ALB' 'CVO' 'APN' 'AHN' 'MMI' 'AUO' 'IN1' 'AUS' 'BSM' 'TX6' 'BGR' 'BIH'
 'BOS' 'BYI' 'CPR' 'CVN' 'CVS' 'DFW' 'DAL' 'FTW' 'FWH' 'AFW' 'ADS' 'DNE'
 'DNN' 'DAY' 'MGY' 'FFO' 'DHN' 'DLH' 'ESN' 'ELM' 'EUG' 'ACV' 'EKA' 'NFL'
 'FAT' 'FCH' 'GUP' 'HLN' '1B1' 'ITH' 'JLN' 'JNU' 'JSE' 'ADQ' 'KDK' 'OKK'
 'LRD' 'LUL' 'LAW' 'LOZ' 'MAE' 'MWA' 'MZZ' 'MNN' 'MCE' 'MOB' 'BFM' 'MLU'
 'MIE' 'APF' 'EWR' 'OXR' 'PIA' 'PIR' 'PUB' 'UIN' 'RAC' 'RSN' 'SLN' 'SRC'
 'NC3' 'SSC' 'SUM' 'TCM' 'GRF' 'TIW' 'TOL' 'TDZ' 'FOE' 'TUS' 'DMA' 'TUP'
 'UVA' 'VEL' 'CWA' 'STE' 'YKM' 'ABI' 'DYS' 'AOO' 'AWX' 'ADM' 'AST' 'ATL'
 'PDK' 'FTY' 'AGS' 'AUG' 'WVL' 'BKW' 'BFR' 'BJI' 'VWL' 'BZN' 'BKG' 'BUF'
 'ORD' 'MDW' 'CGX' 'PWK' 'DPA' 'II2' 'CHI' 'CWI' 'DCU' 'DEC' 'DRT' 'DLF'
 'DTW' 'DET' 'YIP' 'DBQ' 'DRO' 'AMK' 'ELP' 'BIF' 'EKI' 'FDY' 'FET' 'GAD'
 'MS1' 'VWD' 'HKY' 'HLM' 'MI2' 'HOU' 'IAH' 'EFD' 'DWH' 'IDI' 'JXN' 'JAN'
 'HKS' 'MKL' 'JAC' 'OGG' 'EAR' 'GRK' 'ILE' 'ISO' 'LCI' 'LAN' 'LAR' 'LJY'
 'LBL' 'LNK' 'LBB' 'MSN' 'MKT' 'MFR' 'MEM' 'NQA' 'MAF' 'MDD' 'VWH' 'MOD'
 'OLS' 'OFK' 'OAK' 'OLM' 'NY3' 'MCO' 'SFB' 'ORL' 'ISM' 'OSH' 'PAH' 'PHX'
 'SCF' 'DQF' 'DVT' 'LUF' 'AZA' 'PUW' 'RDU' 'RDG' 'RDD' 'O85' 'FVS' 'ROA'
```

```
 'ROW' 'RUI' 'RUT' 'SAD' 'MBS' 'SNS' 'SEA' 'BFI' 'LKE' 'SER' 'SNL' 'SHN'
 'GEG' 'SKA' 'SFF' 'IRS' 'TTN' 'TKF' 'VIS' 'ICT' 'IAB' 'ULS' 'BJJ' 'YKN'
 'ABR' 'AMA' 'TDW' 'AID' 'AND' 'ANB' 'ATW' 'BBC' 'BPT' 'BIL' 'BIS' 'BFD'
 'BRD' 'CAD' 'CNM' 'CYS' 'COU' 'CAE' 'CUB' 'MRC' 'CSG' 'LSF' 'CLU' 'GTR'
 'UBS' 'CBM' 'OLU' 'CMH' 'LCK' 'OSU' 'DNV' 'KY1' 'ESC' 'FLO' 'FEP' 'GCC'
 'GPT' 'HRO' 'BDL' 'HFD' 'HSI' 'HNL' 'HIK' 'IKK' 'EYW' 'NQX' 'LAL' 'LWC'
 'LWS' 'LEW' 'GGG' 'MML' 'MHL' 'ASL' 'MEI' 'MSO' 'MTJ' 'MKG' 'EWN' 'LGA'
 'JFK' 'TSS' 'JRA' 'JRB' 'OWA' 'PWM' 'PDX' 'PRC' 'RID' 'UXJ' 'RIC' 'RIW'
 'RFD' 'RKD' 'RBG' 'SJC' 'SKY' 'SAF' 'SAV' 'SVN' 'AVP' 'SHR' 'SOW' 'SIK'
 'SME' 'SHD' 'STK' 'SQI' 'SCK' 'SYR' 'VLD' 'VAD' 'VCT' 'ALO' 'AYS' 'HLG'
 'ABE' 'ANC' 'EDF' 'MRI' 'ARB' 'AVL' 'BWI' 'MTN' 'XWL' 'PWT' 'BOK' 'BKX'
 'BWD' 'BQK' 'SSI' 'CMI' 'CLT' 'CLE' 'BKL' 'CGF' 'CRS' 'DIK' 'TN6' 'IPL'
 'NJK' 'ELD' 'FAI' 'EIL' 'FBK' 'FLG' 'FFT' 'GBG' 'GSB' 'GRB' 'GWO' 'GRD'
 'JMS' 'JHW' 'JST' 'JBR' 'AZO' 'FCA' 'S27' 'ZXX' 'KTN' 'WFB' 'DQU' 'TYS'
 'LSE' 'LGD' 'LAF' 'LFT' 'XXW' 'LNS' 'LAS' 'LSV' 'VGT' 'HSH' 'TN3' 'LEX'
 'LSD' 'LYH' 'MHK' 'MFD' 'MQT' 'SAW' 'MEJ' 'MKE' 'MWC' 'MUT' 'BNA' 'JWN'
 'HVN' 'OWB' 'PDT' 'PNS' 'NPA' 'PIH' 'RBL' 'RIV' 'RST' 'ROC' 'SBY' 'SRW'
 'SAN' 'NKX' 'NZY' 'SDM' 'MYF' 'SEE' 'SNA' 'NZJ' 'SBM' 'STC' 'STL' 'SUS'
 'CPS' 'ASN' 'THA' 'ART' 'ATY' 'EAT' 'ISN' 'ORH' 'ALM' 'HMN' 'ESF' 'AEX'
 'AXN' 'BVX' 'BLI' 'WBR' 'BGS' 'BGM' 'BHM' 'BDR' 'BRL' 'BTV' 'MDH' 'CDC'
 'CHS' 'CRW' 'WV1' 'CVG' 'LUK' 'OH5' 'CKB' 'CSV' 'CBE' 'DSM' 'DDC' 'EGP'
 'EAU' 'ELN' 'EVV' 'FAM' 'FMN' 'FOD' 'FSM' 'FWA' 'GGE' 'GBD' 'GSO' 'GLH'
 'PGV' 'GSP' 'GDC' 'GMU' 'HGR' 'MDT' 'HSV' 'HUA' 'AL3' 'HUT' 'IRK' 'LRU'
 'LHV' 'LAM' 'MHT' 'MCW' 'MGM' 'MXF' 'MGW' 'MOR' 'MWH' 'ARA' 'ODW' 'OGS'
 'OGB' 'PBF' 'PIT' 'AGC' 'PSF' 'PNC' 'PRZ' 'PVD' 'RAP' 'RCA' 'SMF' 'MHR'
 'MCC' 'SAC' 'SJT' 'STS' 'SHV' 'BAD' 'DTN' 'SUX' 'SBN' 'SGU' 'STJ' 'STF'
 'TBR' 'SWO' 'SLB' 'SWW' 'TCL' 'TWF' 'UCY' 'AXV' 'IAD' 'DCA' 'ILG' 'ILM'
 'ILN' 'WGO' 'YNG' 'ZZV' 'ABQ' 'BFL' 'BTR' 'BMI' 'BMG' 'BYH' 'BRO' 'CHA'
 'OH3' 'CFV' 'GNV' 'GVL' 'GCK' 'GFL' 'GFK' 'GTF' 'GCY' 'PIB' 'HOT' 'IDA'
 'MCI' 'MKC' 'LIT' 'LAX' 'MSP' 'STP' 'FCM' 'LFK' 'MSY' 'NEW' 'NBG' 'PFN'
 'PAM' 'PKB' 'PBG' 'PGD' 'RWI' 'SAT' 'SKF' 'RND' 'BFF' 'SYI' 'SVC' 'FSD'
 'SPA' 'SPI' 'SGF' 'SGH' 'SVH' 'TLH' 'HUF' 'TVI' 'ALW' 'SZL' 'BTL' 'BMC'
 'CID' 'CEV' 'FYV' 'XNA' 'FAY' 'POB' 'FFM' 'GRI' 'GRR' 'IND' 'MQJ' 'JAX'
 'NZC' 'NIP' 'OAJ' 'LCH' 'WQM' 'LWF' 'KY5' 'MIW' 'MVN' 'MVW' 'MYR' 'CRE'
 'LBF' 'PHL' 'PNE' 'CLM' 'POU' 'RKS' 'AR1' 'IPT' 'ACY' 'AIY' 'BWG' 'CEC'
 'EKX' 'IMT' 'LMT' 'WMH' 'MUO' 'OKC' 'TIK' 'PWA' 'SFO' 'JCC' 'SBA' 'SCE'
 'TVC' 'SPS' 'CGI' 'CRP' 'NGP' 'ECG' 'GJT' 'JEF' 'MOP' 'MPS' 'KY3' 'SLC'
 'CHO' 'CLL' 'FLL' 'FXE' 'SBP' 'PRB' 'PBI' 'COS' 'HII' 'PHD' 'TBN' 'OH1']
Unique values in 'Destination_City': ['Ames' 'Bend' 'Elko' 'Enid' 'Erie' 'Ga
ry' 'Hays' 'Hilo' 'Lima' 'Napa'
 'Peru' 'Reno' 'Rome' 'Taos' 'Waco' 'York' 'Yuma' 'Akron' 'Altus' 'Butte'
 'Chico' 'Dover' 'Fargo' 'Flint' 'Havre' 'Hobbs' 'Houma' 'Huron' 'Logan'
 'Macon' 'Miami' 'Minot' 'Ocala' 'Ogden' 'Omaha' 'Paris' 'Price' 'Provo'
 'Salem' 'Selma' 'Tampa' 'Tulsa' 'Tyler' 'Ukiah' 'Utica' 'Albany' 'Alpena'
 'Athens' 'Auburn' 'Austin' 'Bangor' 'Bishop' 'Boston' 'Burley' 'Casper'
 'Clovis' 'Dallas' 'Dalton' 'Dayton' 'Dothan' 'Duluth' 'Easton' 'Elmira'
 'Eugene' 'Eureka' 'Fallon' 'Fresno' 'Gallup' 'Helena' 'Hudson' 'Ithaca'
 'Joplin' 'Juneau' 'Kodiak' 'Kokomo' 'Laredo' 'Laurel' 'Lawton' 'London'
 'Madera' 'Marion' 'Merced' 'Mobile' 'Monroe' 'Muncie' 'Naples' 'Newark'
 'Oxnard' 'Peoria' 'Pierre' 'Pueblo' 'Quincy' 'Racine' 'Ruston' 'Salina'
 'Searcy' 'Shelby' 'Sumter' 'Tacoma' 'Toledo' 'Topeka' 'Tucson' 'Tupelo'
 'Uvalde' 'Vernal' 'Wausau' 'Yakima' 'Abilene' 'Altoona' 'Andrews'
 'Ardmore' 'Astoria' 'Atlanta' 'Augusta' 'Beckley' 'Bedford' 'Bemidji'
```

```
'Bozeman' 'Branson' 'Buffalo' 'Chicago' 'Clinton' 'Decatur' 'Del Rio'
'Detroit' 'Dubuque' 'Durango' 'El Paso' 'Elkhart' 'Findlay' 'Fremont'
'Gadsden' 'Grenada' 'Hanford' 'Hickory' 'Holland' 'Houston' 'Indiana'
'Jackson' 'Kahului' 'Kearney' 'Killeen' 'Kinston' 'Laconia' 'Lansing'
'Laramie' 'Lebanon' 'Liberal' 'Lincoln' 'Lubbock' 'Madison' 'Mankato'
'Medford' 'Memphis' 'Midland' 'Modesto' 'Nogales' 'Norfolk' 'Oakland'
'Olympia' 'Oneonta' 'Orlando' 'Oshkosh' 'Paducah' 'Phoenix' 'Pullman'
'Raleigh' 'Reading' 'Redding' 'Rexburg' 'Roanoke' 'Roswell' 'Ruidoso'
'Rutland' 'Safford' 'Saginaw' 'Salinas' 'Seattle' 'Seymour' 'Shawnee'
'Shelton' 'Spokane' 'Sturgis' 'Trenton' 'Truckee' 'Visalia' 'Wichita'
'Wooster' 'Yankton' 'Aberdeen' 'Amarillo' 'Anderson' 'Anniston'
'Appleton' 'Bay City' 'Beaumont' 'Billings' 'Bismarck' 'Bradford'
'Brainerd' 'Cadillac' 'Carlsbad' 'Cheyenne' 'Columbia' 'Columbus'
'Danville' 'Escanaba' 'Florence' 'Freeport' 'Gillette' 'Gulfport'
'Harrison' 'Hartford' 'Hastings' 'Honolulu' 'Kankakee' 'Key West'
'Lakeland' 'Lawrence' 'Lewiston' 'Longview' 'Marshall' 'Meridian'
'Missoula' 'Montrose' 'Muskegon' 'New Bern' 'New York' 'Owatonna'
'Portland' 'Prescott' 'Richmond' 'Riverton' 'Rockford' 'Rockland'
'Roseburg' 'San Jose' 'Sandusky' 'Santa Fe' 'Savannah' 'Scranton'
'Sheridan' 'Show Low' 'Sikeston' 'Somerset' 'Staunton' 'Sterling'
'Stockton' 'Syracuse' 'Valdosta' 'Victoria' 'Waterloo' 'Waycross'
'Wheeling' 'Allentown' 'Anchorage' 'Ann Arbor' 'Asheville' 'Baltimore'
'Blackfoot' 'Bremerton' 'Brookings' 'Brownwood' 'Brunswick' 'Champaign'
'Charlotte' 'Cleveland' 'Corsicana' 'Dickinson' 'Dyersburg' 'El Centro'
'El Dorado' 'Fairbanks' 'Flagstaff' 'Frankfort' 'Galesburg' 'Goldsboro'
'Green Bay' 'Greenwood' 'Jamestown' 'Johnstown' 'Jonesboro' 'Kalamazoo'
'Kalispell' 'Kennewick' 'Ketchikan' 'Knoxville' 'La Crosse' 'La Grande'
'Lafayette' 'Lake City' 'Lancaster' 'Las Vegas' 'Lewisburg' 'Lexington'
'Lynchburg' 'Manhattan' 'Mansfield' 'Marquette' 'Meadville' 'Milwaukee'
'Muscatine' 'Nashville' 'New Haven' 'Owensboro' 'Pendleton' 'Pensacola'
'Pocatello' 'Red Bluff' 'Riverside' 'Rochester' 'Salisbury' 'San Diego'
'Santa Ana' 'Sheboygan' 'St. Cloud' 'St. Louis' 'Talladega' 'Tullahoma'
'Watertown' 'Wenatchee' 'Williston' 'Worcester' 'Alamogordo' 'Alexandria'
'Batesville' 'Bellingham' 'Big Rapids' 'Big Spring' 'Binghamton'
'Birmingham' 'Bridgeport' 'Burlington' 'Carbondale' 'Cedar City'
'Charleston' 'Cincinnati' 'Clarksburg' 'Crossville' 'Cumberland'
'Des Moines' 'Dodge City' 'Eagle Pass' 'Eau Claire' 'Ellensburg'
'Evansville' 'Farmington' 'Fort Dodge' 'Fort Smith' 'Fort Wayne'
'Georgetown' 'Great Bend' 'Greensboro' 'Greenville' 'Hagerstown'
'Harrisburg' 'Huntsville' 'Hutchinson' 'Kirksville' 'Las Cruces'
'Lock Haven' 'Los Alamos' 'Manchester' 'Mason City' 'Montgomery'
'Morgantown' 'Morristown' 'Moses Lake' 'New Iberia' 'Oak Harbor'
'Ogdensburg' 'Orangeburg' 'Pine Bluff' 'Pittsburgh' 'Pittsfield'
'Ponca City' 'Prineville' 'Providence' 'Rapid City' 'Sacramento'
'San Angelo' 'Santa Rosa' 'Shreveport' 'Sioux City' 'South Bend'
'St. George' 'St. Joseph' 'Starkville' 'Statesboro' 'Stillwater'
'Storm Lake' 'Sweetwater' 'Tuscaloosa' 'Twin Falls' 'Union City'
'Wapakoneta' 'Washington' 'Wilmington' 'Winchester' 'Youngstown'
'Zanesville' 'Albuquerque' 'Bakersfield' 'Baton Rouge' 'Bloomington'
'Blytheville' 'Brownsville' 'Chattanooga' 'Chillicothe' 'Coffeyville'
'Gainesville' 'Garden City' 'Glens Falls' 'Grand Forks' 'Great Falls'
'Greeneville' 'Hattiesburg' 'Hot Springs' 'Idaho Falls' 'Kansas City'
'Little Rock' 'Los Angeles' 'Minneapolis' 'Nacogdoches' 'New Orleans'
```

```
 'Panama City' 'Parkersburg' 'Plattsburgh' 'Punta Gorda' 'Rocky Mount'
 'San Antonio' 'Scottsbluff' 'Shelbyville' 'Silver City' 'Sioux Falls'
 'Spartanburg' 'Springfield' 'Statesville' 'Tallahassee' 'Terre Haute'
 'Thomasville' 'Walla Walla' 'Warrensburg' 'Battle Creek' 'Brigham City'
 'Cedar Rapids' 'Connersville' 'Fayetteville' 'Fergus Falls'
 'Grand Island' 'Grand Rapids' 'Indianapolis' 'Jacksonville'
 'Lake Charles' 'Lawrenceburg' 'Madisonville' 'Marshalltown'
 'Mount Vernon' 'Myrtle Beach' 'North Platte' 'Philadelphia'
 'Port Angeles' 'Poughkeepsie' 'Rock Springs' 'Russellville'
 'Williamsport' 'Atlantic City' 'Bowling Green' 'Crescent City'
 'Elizabethtown' 'Iron Mountain' 'Klamath Falls' 'Mountain Home'
 'Oklahoma City' 'San Francisco' 'Santa Barbara' 'State College'
 'Traverse City' 'Wichita Falls' 'Cape Girardeau' 'Corpus Christi'
 'Elizabeth City' 'Grand Junction' 'Jefferson City' 'Mount Pleasant'
 'Mount Sterling' 'Salt Lake City' 'Charlottesville' 'College Station'
 'Fort Lauderdale' 'San Luis Obispo' 'West Palm Beach' 'Colorado Springs'
 'Lake Havasu City' 'New Philadelphia' 'Fort Leonard Wood'
 'Washington Court House']
Unique values in 'Destination_State': ['IA' 'OR' 'NV' 'OK' 'PA' 'IN' 'KS' 'H
I' 'OH' 'CA' 'GA' 'NM' 'TX' 'AZ'
 'MT' 'DE' 'ND' 'MI' 'LA' 'SD' 'UT' 'FL' 'NE' 'TN' 'AL' 'NY' 'ME' 'MA'
 'ID' 'WY' 'MN' 'MD' 'MO' 'AK' 'MS' 'KY' 'IL' 'NJ' 'CO' 'WI' 'AR' 'NC'
 'SC' 'WA' 'WV' 'NH' 'VA' 'VT' 'CT' 'RI' 'DC']
```

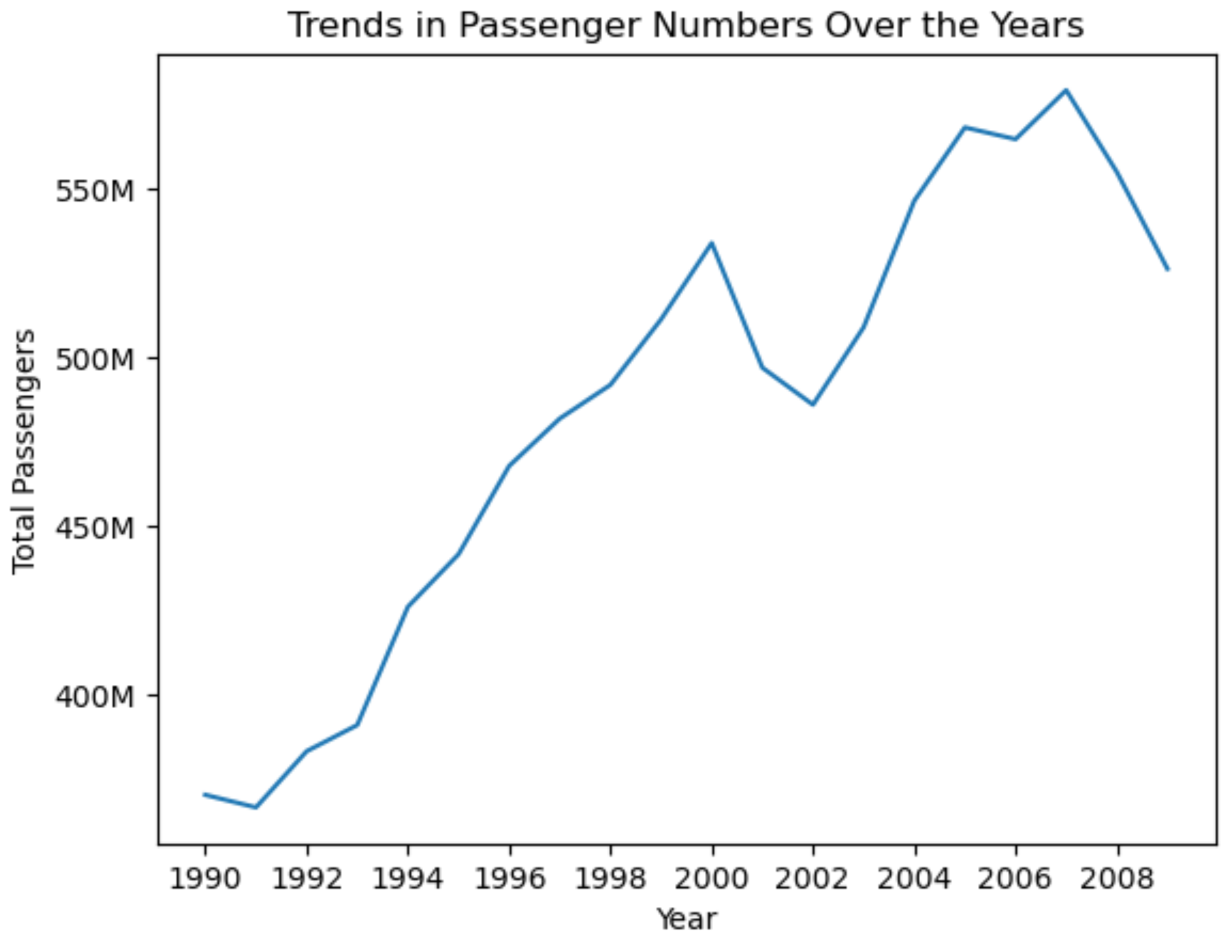
Inference

- Various analysis were performed in the next stages to derive better conclusions from the datas.

Passenger Trends

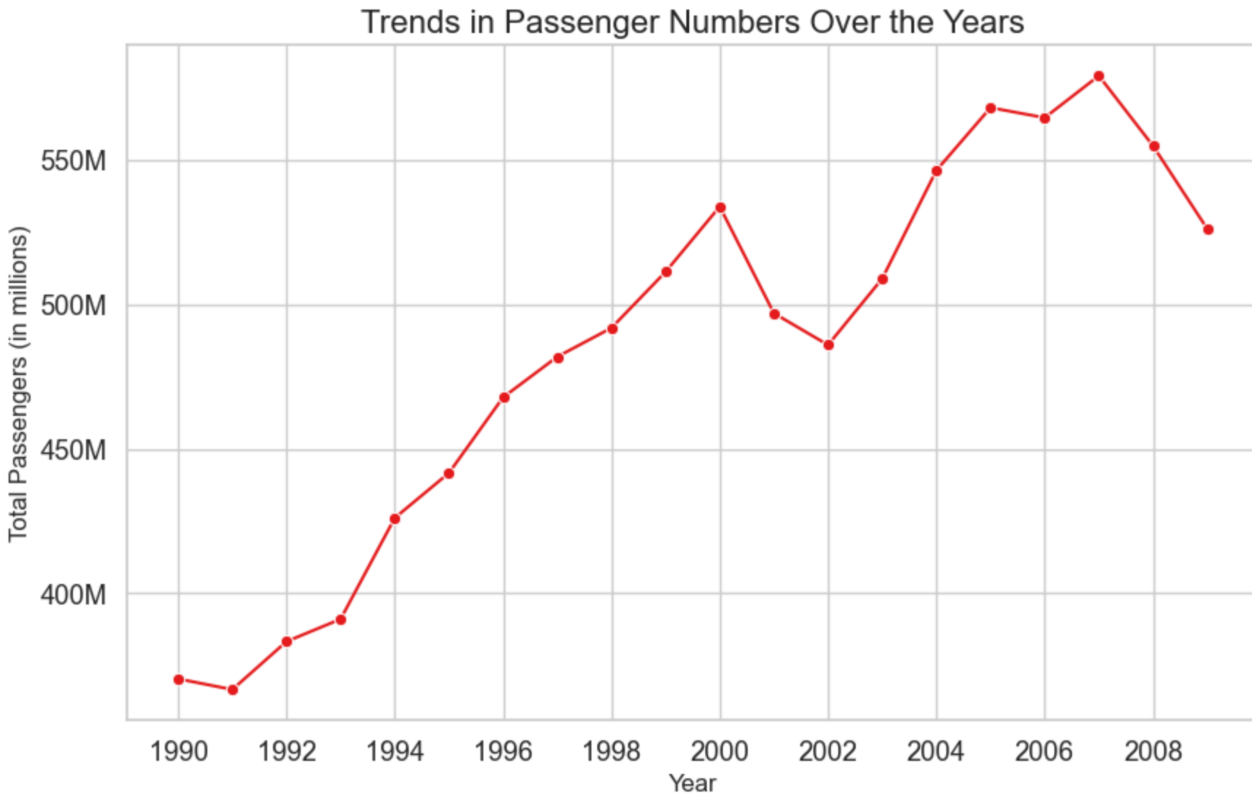
- 'Year' column in x-axis and 'Passengers' column in y-axis
- Let's set the x-axis ticks to show integers from 1990 to 2009 with a step of 2
- Let's format y-axis labels in millions

In [8]:
```
tasks.passenger_trends()
```



- For the Seaborn, let's adjust the figure size as 10X6 for clear picture.
- ***Based on the graphs shown above and below, we can cleary see that there is a drastic drop in flight passengers from 2000 to 2002. The main reason for this could be the 9/11 attact that has occured in the year 2001 which affected US commercial airways to a great extent.***

In [9]:
```
tasks.passenger_trends_seaborn()
```



Top Flight Routes

- Let's combine Origin and Destination columns to create a 'Route' column as follows. The top flight route is from Kahului OGG to Honolulu HNL in Hawai.

In [10]:
```
tasks.top_routes()
```

```
Route
OGG to HNL    32364612
HNL to OGG    29744742
LAX to HNL    28964154
HNL to LAX    28632161
LAS to LAX    26333721
                ...
LBL to PIA           0
LBL to RFD           0
LBL to SDM           0
LBL to YIP           0
ZZV to YIP           0
Name: Passengers, Length: 36719, dtype: int64
```

- As shown below, for city routes, we can clearly see most of the passengers flew in Texas and between Dallas and Houston and the reason behind this could be the Texas being highest populated and vast state.

In [11]:
```
tasks.top_city_routes()
```

```
City_Route
Dallas to Houston           38295025
Houston to Dallas           37989016
Kahului to Honolulu         32364664
Honolulu to Kahului         29744742
Los Angeles to Honolulu     28964232
                              ...
Laredo to Asheville                0
Laredo to Athens                   0
Laredo to Atlanta                  0
Laredo to Auburn                   0
Zanesville to Shreveport           0
Name: Passengers, Length: 28326, dtype: int64
```

- As shown below, for state routes, ***we can see that highest number of the passengers flew in Texas and California states and the reason behind this could be that those two states are highest populated and vast states in the Unites States after Alaska.***

In [12]:
```
tasks.top_state_routes()
```

```
State_Route
TX to TX    338568795
CA to CA    272055082
GA to FL    109493247
FL to GA    108403701
TX to CA     96358405
              ...
NH to MT            0
ID to MD            0
AL to ND            0
ME to AK            0
DE to NJ            0
Name: Passengers, Length: 2422, dtype: int64
```

Monthly Passengers

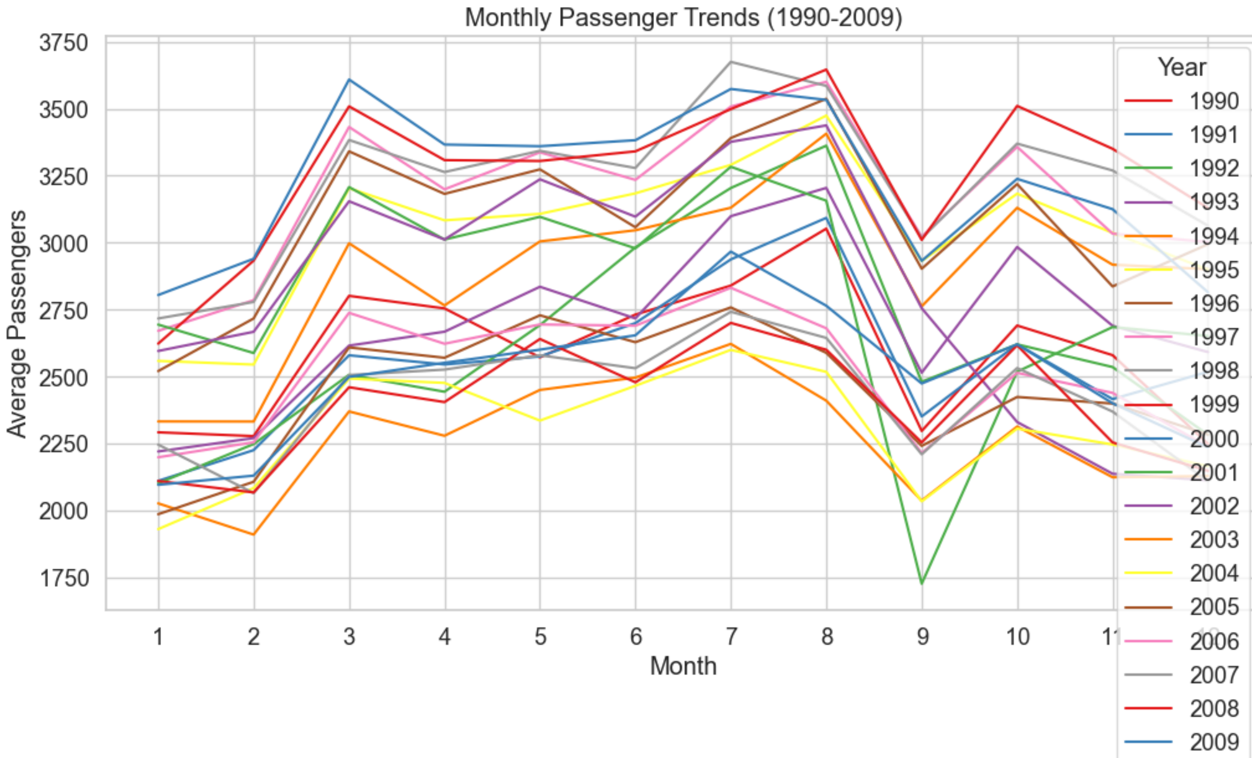
- To find out which months of the year, passengers are mostly using flights. Let's calculate the mean of passengers by usnig groupby function over months of all years.
- As shown below, we can clearly see that most usage of flights is during the months of July and August and least during January and February.
- ***The reason behind the most usage could be the summer vacation, holidays, events and festivals during june, july and august, where as the least because of unfavourable cold weather conditions for tourism during january and february.***

In [13]:
```
tasks.monthly_passengers()
```

Out[13]:
```
Month
7     3052.8
8     3032.6
5     2848.4
3     2846.4
6     2841.6
4     2761.5
10    2743.9
11    2585.8
12    2461.9
9     2454.5
2     2352.2
1     2306.5
Name: Passengers, dtype: float64
```

- Let's plot this monthly passengers trend in different line colours during each and every year in the dataset by using subplots and for loop. As shown below, we can clearly see that the mean monthly passengers has given us the nearest result for the conclusions that were derived earlier.

In [14]:
```
tasks.monthly_passengers_plot()
```



- As shown below, we can clearly see that it is more clear in the seaborn plot as it easy to read the data and arrive at decisions.

In [15]:
```
tasks.monthly_passengers_seaborn()
```

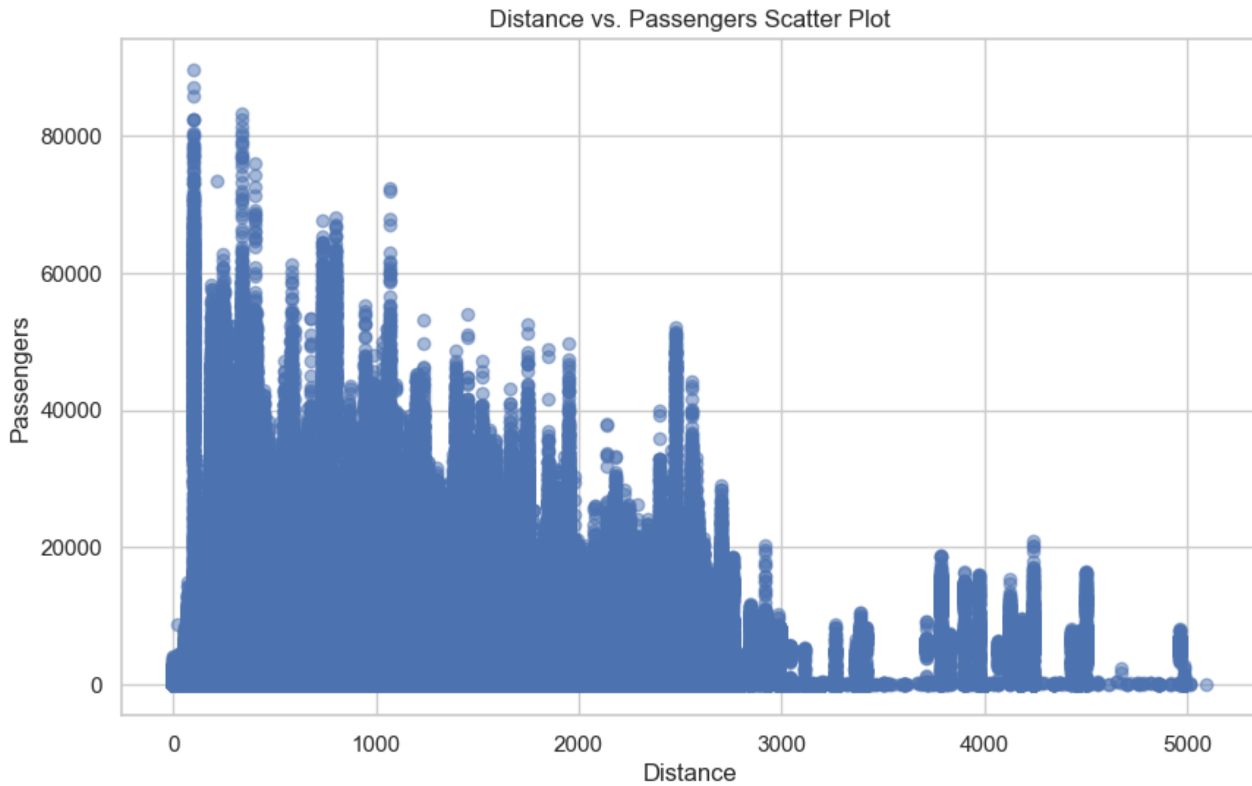
- Let's find out which route in the Unites States has more demand by counting the number of flights in a particular route.
- As shown below, we can clearly conclude that the busiest route is from Los Angeles LAX to Sanfrancisco SFO and Hawai HNL

In [16]:
```
tasks.flights_per_route()
```

Out[16]:
```
Route
LAX to SFO    5694
LAX to HNL    5510
SFO to LAX    4767
HNL to LAX    4753
DTW to ORD    4250
              ...
IN1 to SYR       1
RNO to SYR       1
GMU to SYR       1
LAR to SYR       1
FWA to OH1       1
Name: count, Length: 36719, dtype: int64
```

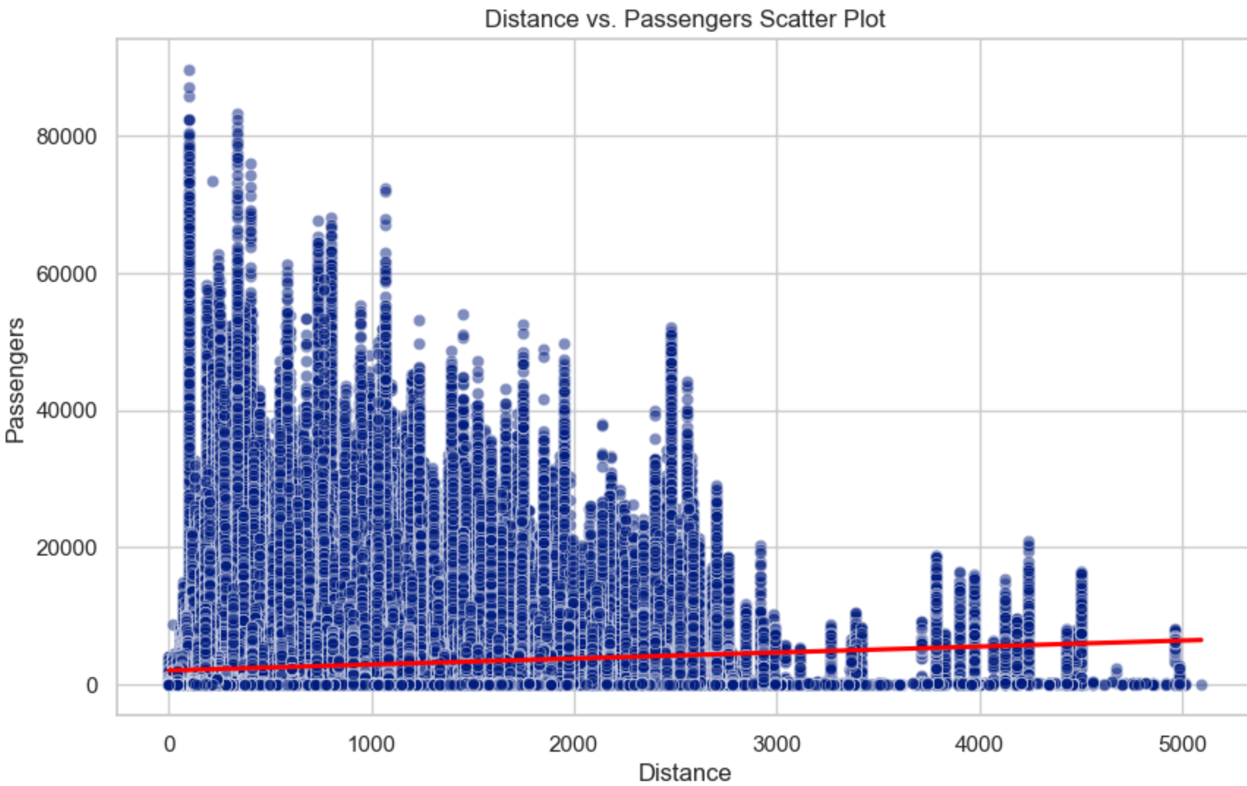
- Let's analyze the distance for which the passengers are opting for flights accross United states by plotting a graph between distance and their corresponding sum of passengers travelling.

In [17]:
```
tasks.distance_vs_passengers()
```



- As shown below, ***we can conclude that flights were used the most for upto 1100 miles and there after there is a gradual drop in flights usage over increasing distance. We can also conclude that there is a significant drop after 2700-2800 miles which can be clearly seen in seaborn plot comparatively.***
- ***This conclusion stands on par with the longest line in america by Dr. Cliff Pickover which is 2802 miles as shown in his work.***
- ***2802miles is the longest distance connecting 2 extreme points in USA excluding Alaska and the increasing trend after nearly 3800 miles might be due to the crossing distance over Canada to Alaska.***

In [19]:
```
tasks.distance_vs_passengers_seaborn()
```



- As shown below, we can also find the most busiest airports and the least ones by summing up the passengers arriving and departing from the corresponding airports accross the USA. It is found that Atlanta ATL airport is one of the most busiest airports in USA where as few airports which does not have a single passenger which is basically not possible, the reason possibly behind this could be inadequate data.

In [20]:
```
tasks.airports()
```

```
Top Airports:
Airport
ATL    577124268
ORD    529018110
DFW    457153720
LAX    393005676
PHX    295857703
LAS    270590248
DTW    250983023
MSP    245197238
SFO    243779917
IAH    228367851
Name: Passengers, dtype: int64

Bottom Airports:
Airport
ZZV    0
BYH    0
II2    0
IKK    0
IN1    0
BVX    0
IRS    0
ISM    0
JRA    0
JXN    0
Name: Passengers, dtype: int64
```

- As shown below, the passengers data as per the state was extracted by groupby function over the origin states and summing up the corresponding passengers over the years. Most people in the USA who uses flights are from California and Texas states followed by Florida and Delaware state people uses flights to the least.
- There might be various reasons like unavailability of airways in that state becuase of geographical reasons.

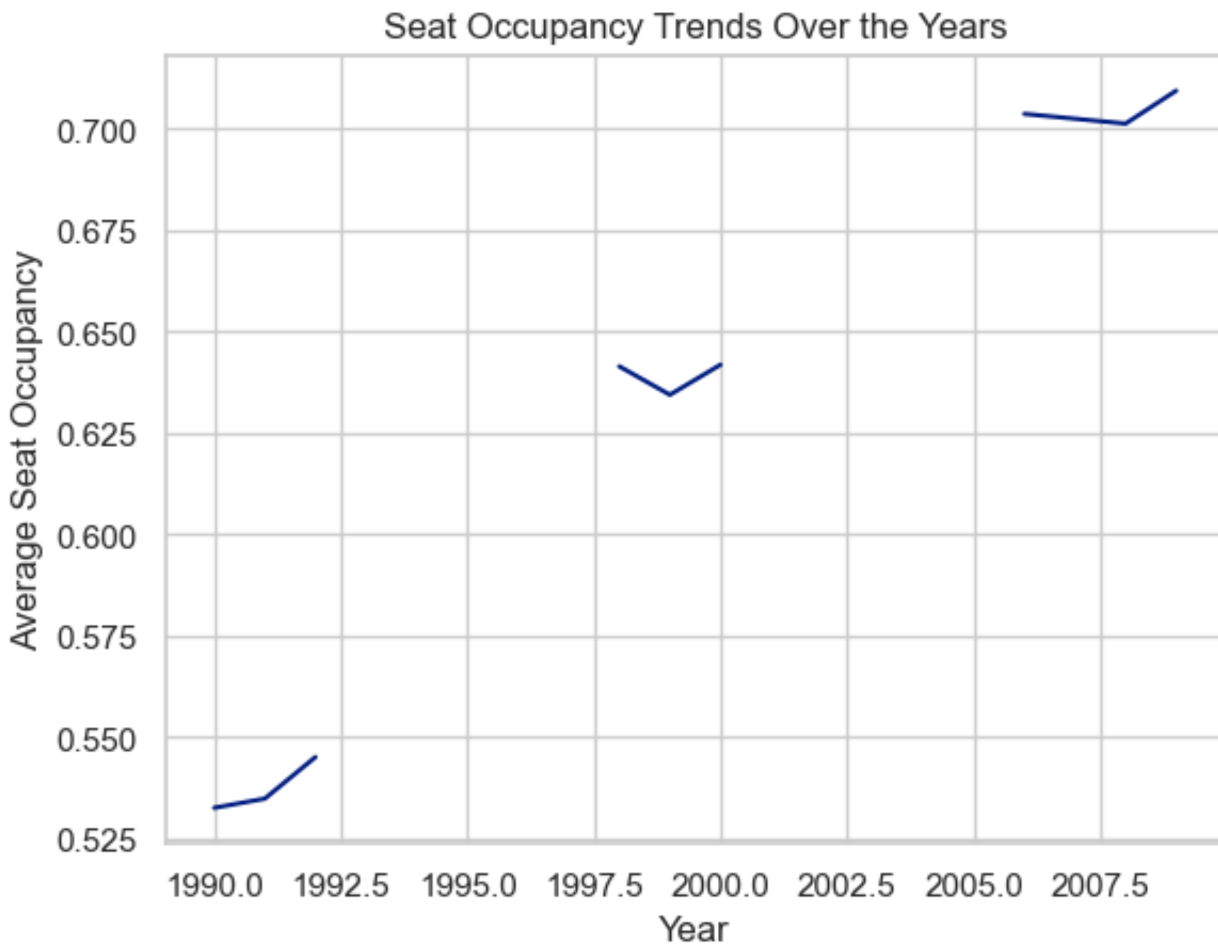
In [21]:
```
tasks.state_passenger_data()
```

```
Top States by Passenger Count:
Origin_State
CA    1078576915
TX    1065234417
FL     756964875
IL     658813004
GA     598611798
NY     448222654
PA     353921034
AZ     329206856
OH     325641266
NC     320445210
NV     318422665
```

```
MO     309263535
MI     292708282
DC     252426311
MN     251824115
WA     243174421
NJ     215271955
TN     191268090
MA     190295148
HI     179129866
UT     152475194
MD     143300416
OR     112979688
LA     106031996
IN      81111481
WI      80267300
OK      64342623
NM      60463602
CT      55092210
AL      50812671
SC      49747994
AK      47516451
RI      38759528
NE      32268245
VA      31401623
AR      30543812
IA      23726479
MT      23662987
NH      22610638
MS      19158008
CO      18612934
ME      16595592
KS      11090609
ND       9851185
KY       8972636
VT       8681885
SD       8195983
WV       4249010
ID       3352777
WY       3043146
DE         29097
Name: Passengers, dtype: int64
```

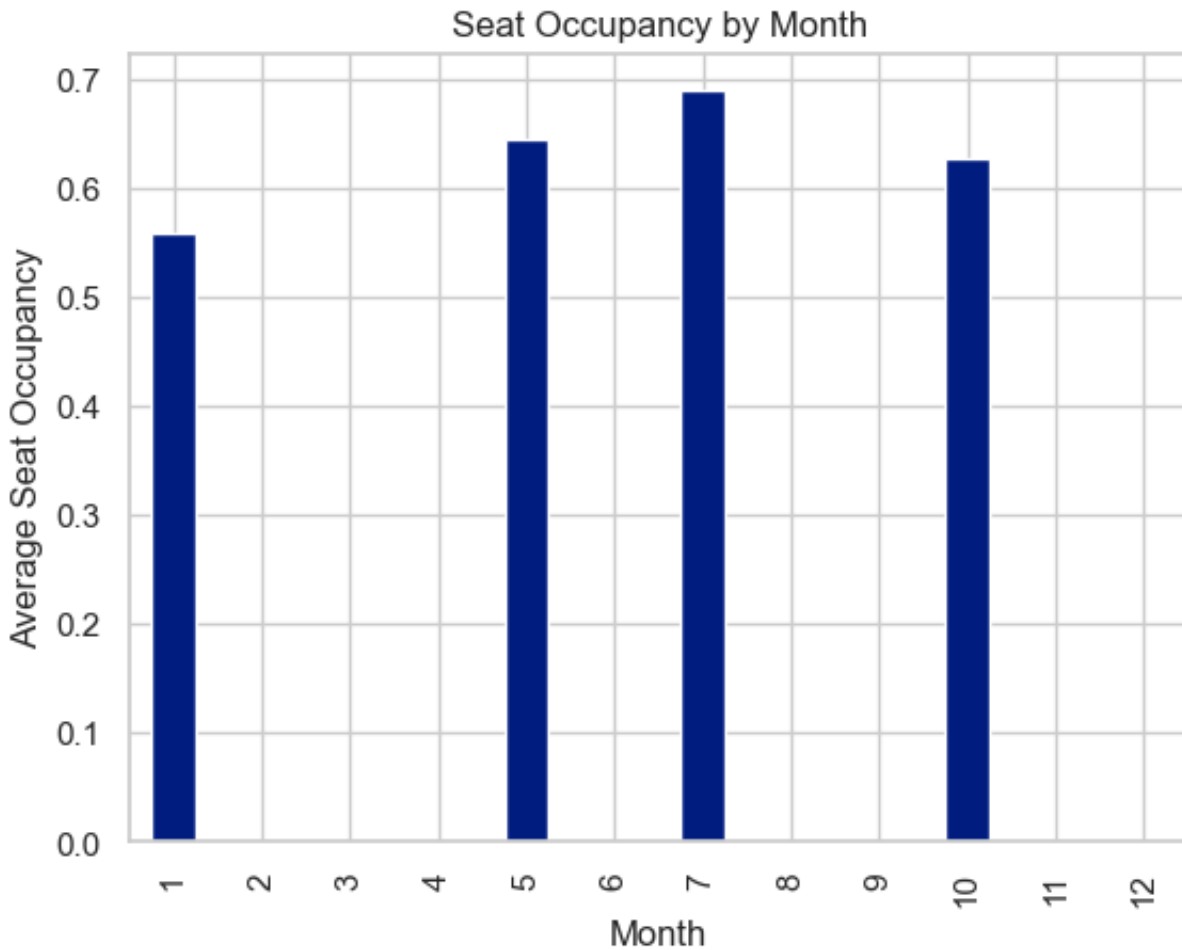
- Seat Occupancy is also one of the measures we can use to examine if a flight is making money and if it is efficient in terms of transportation. Let's calculate the seat occupancy by dividing number of passengers with the number of seats. If it's value is 1 then it means that the flight is completely filled and any value between 0 and 1 is for partially filled.
- As shown below, we can see that there are few breaks in the data plot which might be due to missing values or inaccurate data.

In [22]:
```
tasks.seat_occupancy_year()
```



- As shown below, we can see that there are few breaks in the data plot which might be due to missing values or inaccurate data.

In [23]:
```
tasks.seat_occupancy_month()
```



- With the help of seat occupancy on specific routes it is easy to find out the routes in which the flight is completely booked. As shown below, for the flight route from TTN-Trenton to PHL-Philadelphia, we can see that the seat occupancy is 1.5 which means that the flight is over booked, it is clearly not possible for a fixed number of seats, it could be due to wrong data row which is having more number of passengers than number of available seats.
- We can see that there are many flight routes which are completely booked such as RDU-Raleigh to PKB-Parkersburg and Oklahoma to Lawrence (probably due to oil and gas companies)

In [24]:
```
tasks.route_seat_occupancy()
```

```
Average Seat Occupancy by Route:
Route
TTN to PHL   1.5
RDU to PKB   1.0
OKC to LWC   1.0
TUL to BUF   1.0
LBF to PIA   1.0
            ..
YUM to YIP   NaN
ZZV to MCI   NaN
ZZV to PDK   NaN
ZZV to SHV   NaN
ZZV to YIP   NaN
Name: Seat_Occupancy, Length: 36719, dtype: float64
```

- Let's examine each flight's Seat Occupancy closely, to find out the exact reason behind the breaks in the plots. We can see the infinite values over here

In [25]:
```
tasks.Seat_Occupancy()
```

```
Rows Sorted by Seat Occupancy:
        Origin       Origin_City Origin_State Destination  \
2801180    BKW           Beckley           WV         IAD
700161     IAD        Washington           DC         BKW
2119807    ANC         Anchorage           AK         FAI
700183     IAD        Washington           DC         BKW
1695204    MIA             Miami           FL         JFK
...        ...               ...          ...         ...
3606572    PHX           Phoenix           AZ         HII
3606573    PHX           Phoenix           AZ         HII
3606574    YUM              Yuma           AZ         HII
3606575    HII  Lake Havasu City           AZ         HII
3606802    FWA        Fort Wayne           IN         OH1

                 Destination_City Destination_State  Year  Month  Passengers
\
2801180                Washington                DC  2002     12           7
700161                    Beckley                WV  2002     12           2
2119807                 Fairbanks                AK  2001      8           1
700183                    Beckley                WV  2003      3           2
1695204                  New York                NY  1995      9        1367
...                           ...               ...   ...    ...         ...
3606572          Lake Havasu City                AZ  2009      3           0
3606573          Lake Havasu City                AZ  2009      1           0
3606574          Lake Havasu City                AZ  2009      5           0
3606575          Lake Havasu City                AZ  2009      2           0
3606802   Washington Court House                OH  2003      9           0

         Seats  Flights  Distance  Origin Population  Destination Population
\
2801180      0        0     215.0              78851                10029142
700161       0        0     215.0           10029142                   78851
```

```
2119807      0       16     261.0              325839                  85233
700183       0        0     215.0            10172752                  78587
1695204      0       12    1090.0             8913928               34261384
...        ...      ...       ...                 ...                    ...
3606572      0       21     156.0             4364094                 194825
3606573      0       21     156.0             4364094                 194825
3606574      0        1     133.0              196972                 194825
3606575      0        1       0.0              194825                 194825
3606802      0        1     135.0              398574                  28133

              Route                            City_Route State_Route Airpor
t  \
2801180  BKW to IAD                 Beckley to Washington    WV to DC     BK
W
700161   IAD to BKW                 Washington to Beckley    DC to WV     IA
D
2119807  ANC to FAI                Anchorage to Fairbanks    AK to AK     AN
C
700183   IAD to BKW                 Washington to Beckley    DC to WV     IA
D
1695204  MIA to JFK                     Miami to New York    FL to NY     MI
A
...             ...                                   ...         ...       
...
3606572  PHX to HII          Phoenix to Lake Havasu City    AZ to AZ     PH
X
3606573  PHX to HII          Phoenix to Lake Havasu City    AZ to AZ     PH
X
3606574  YUM to HII             Yuma to Lake Havasu City    AZ to AZ     YU
M
3606575  HII to HII  Lake Havasu City to Lake Havasu City    AZ to AZ     HI
I
3606802  FWA to OH1  Fort Wayne to Washington Court House    IN to OH     FW
A

         Seat_Occupancy
2801180             inf
700161              inf
2119807             inf
700183              inf
1695204             inf
...                 ...
3606572             NaN
3606573             NaN
3606574             NaN
3606575             NaN
3606802             NaN

[3606803 rows x 19 columns]
```

- As shown below, it is beacause of the zero values being assigned to seats in few rows of data leading to infinite values for Seat Occupancy.

In [26]:
```
tasks.Examine_Seat_Occupancy()
```

```
Details of Row 2801180:
Origin                                          BKW
Origin_City                                 Beckley
Origin_State                                     WV
Destination                                     IAD
Destination_City                         Washington
Destination_State                                DC
Year                                           2002
Month                                            12
Passengers                                        7
Seats                                             0
Flights                                           0
Distance                                      215.0
Origin Population                             78851
Destination Population                     10029142
Route                                    BKW to IAD
City_Route                    Beckley to Washington
State_Route                                WV to DC
Airport                                         BKW
Seat_Occupancy                                  inf
Name: 2801180, dtype: object
Number of Rows with Zero Seats: 334036
```

- Although it is not ideal to replace zero Seats with their corresponding Passengers, let's try this to know if this is the main reason for few breaks in the plot. As shown below, the zero seats are replaced by their corresponding passengers, 7 in this case.

In [27]:
```
tasks.Modify_Seat_Occupancy()
```

```
Details of Row 2801180 after Modification:
Origin                                       BKW
Origin_City                              Beckley
Origin_State                                  WV
Destination                                  IAD
Destination_City                      Washington
Destination_State                             DC
Year                                        2002
Month                                         12
Passengers                                     7
Seats                                          7
Flights                                        0
Distance                                   215.0
Origin Population                          78851
Destination Population                  10029142
Route                                 BKW to IAD
City_Route             Beckley to Washington
State_Route                             WV to DC
Airport                                      BKW
Seat_Occupancy                               inf
Name: 2801180, dtype: object
```

- Let's calculate the new seat occupancy by dividing number of passengers with the number of seats.

In [29]:
```
tasks.Seat_Occupancy_new()
```

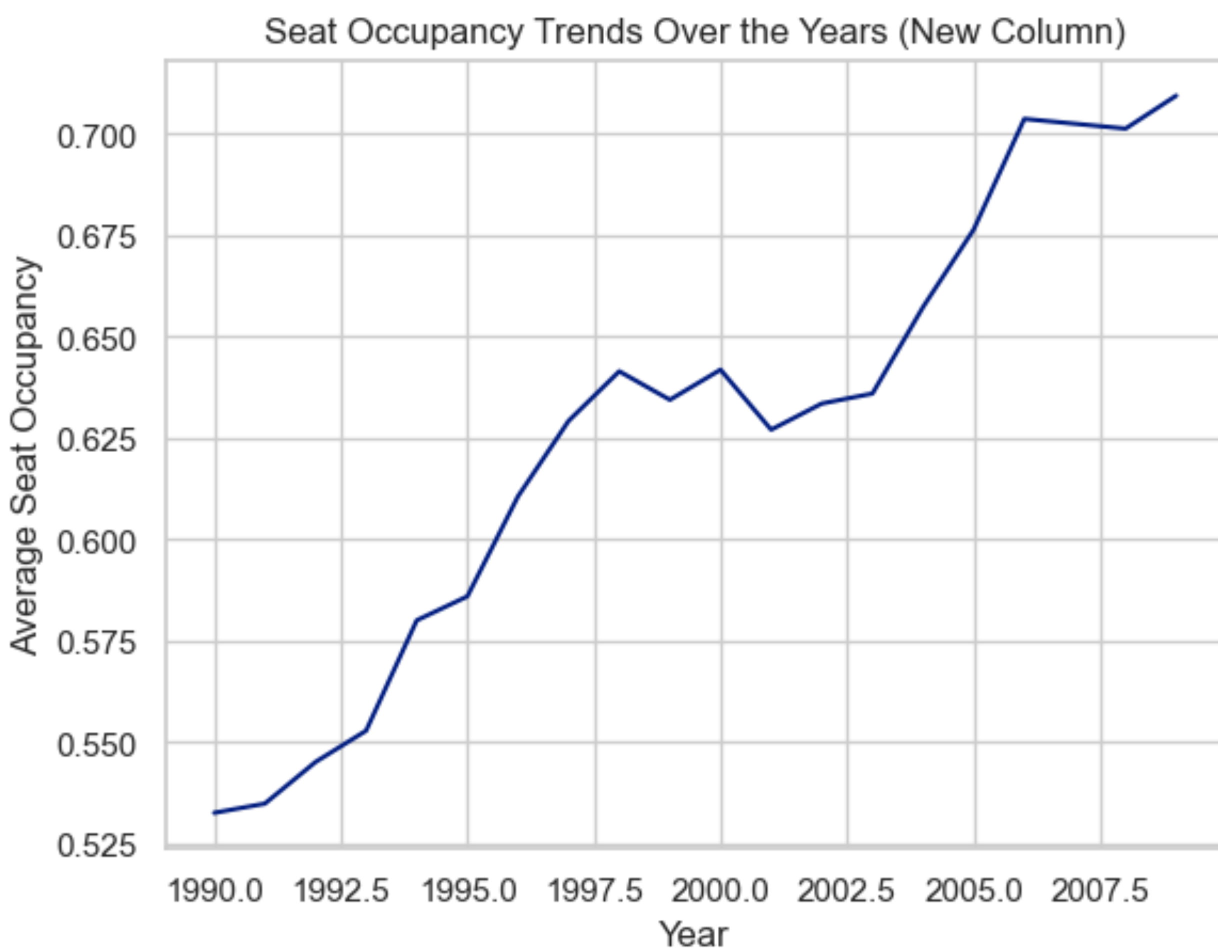
```
Rows Sorted by Seat Occupancy (New Column):
        Origin       Origin_City Origin_State Destination  \
3341600    TTN           Trenton           NJ         PHL
1898495    SEA           Seattle           WA         ANC
2165887    DTW           Detroit           MI         LAS
1040966    ATL           Atlanta           GA         IAH
1287123    FLL   Fort Lauderdale           FL         MCO
...        ...               ...          ...         ...
3606572    PHX           Phoenix           AZ         HII
3606573    PHX           Phoenix           AZ         HII
3606574    YUM              Yuma           AZ         HII
3606575    HII  Lake Havasu City           AZ         HII
3606802    FWA        Fort Wayne           IN         OH1

               Destination_City Destination_State  Year  Month  Passengers
\
3341600            Philadelphia                PA  1998     11        1349
1898495               Anchorage                AK  1992      2         462
2165887               Las Vegas                NV  1994      1         469
1040966                 Houston                TX  1990      9         553
1287123                 Orlando                FL  1990      5        1205
...                         ...               ...   ...    ...         ...
3606572        Lake Havasu City                AZ  2009      3           0
3606573        Lake Havasu City                AZ  2009      1           0
3606574        Lake Havasu City                AZ  2009      5           0
```

```
3606575        Lake Havasu City                 AZ  2009      2           0
3606802  Washington Court House                 OH  2003      9           0

         Seats  Flights  Distance  Origin Population  Destination Population
\
3341600    122        3      37.0             331474                11020546
1898495    233        1    1448.0            5347562                  289910
2165887    256        1    1750.0            8761684                  938611
1040966    360        1     689.0            3087755                 3789490
1287123    792        4     178.0            4074690                 1239115
...        ...      ...       ...                ...                     ...
3606572      0       21     156.0            4364094                  194825
3606573      0       21     156.0            4364094                  194825
3606574      0        1     133.0             196972                  194825
3606575      0        1       0.0             194825                  194825
3606802      0        1     135.0             398574                   28133

              Route                               City_Route State_Route Airpor
t  \
3341600  TTN to PHL                Trenton to Philadelphia    NJ to PA     TT
N
1898495  SEA to ANC                   Seattle to Anchorage    WA to AK     SE
A
2165887  DTW to LAS                   Detroit to Las Vegas    MI to NV     DT
W
1040966  ATL to IAH                    Atlanta to Houston    GA to TX     AT
L
1287123  FLL to MCO           Fort Lauderdale to Orlando    FL to FL     FL
L
...             ...                                  ...         ...
...
3606572  PHX to HII           Phoenix to Lake Havasu City    AZ to AZ     PH
X
3606573  PHX to HII           Phoenix to Lake Havasu City    AZ to AZ     PH
X
3606574  YUM to HII              Yuma to Lake Havasu City    AZ to AZ     YU
M
3606575  HII to HII  Lake Havasu City to Lake Havasu City    AZ to AZ     HI
I
3606802  FWA to OH1  Fort Wayne to Washington Court House    IN to OH     FW
A

         Seat_Occupancy  Seat_Occupancy_new
3341600            11.1                11.1
1898495             2.0                 2.0
2165887             1.8                 1.8
1040966             1.5                 1.5
1287123             1.5                 1.5
...                 ...                 ...
3606572             NaN                 NaN
3606573             NaN                 NaN
3606574             NaN                 NaN
3606575             NaN                 NaN
```

```
3606802              NaN                    NaN

[3606803 rows x 20 columns]
```

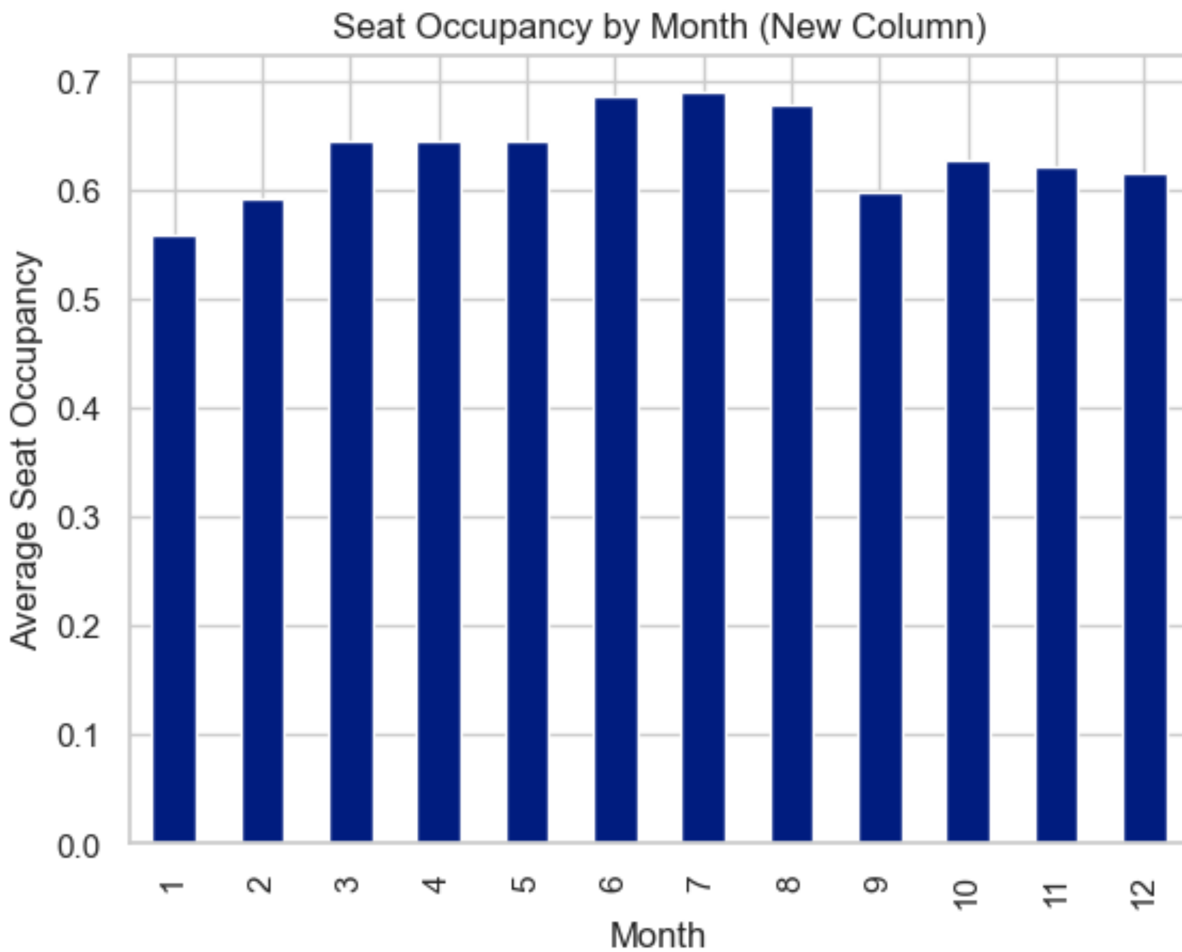
- As shown below, ***we can clearly see a continuous graph line making a meaningful depiction of the data correection that has been made. We can conclude that there is an increasing trend in seat occupancy from 1990 to 2009 except during 2001(due to 9/11 attack). This could have been only possible by employing various data anaytic measures by various airline companies in designing their flight routes for maximizing their profits over the years.***

In [30]:
```
tasks.Seat_Occupancy_new_year()
```



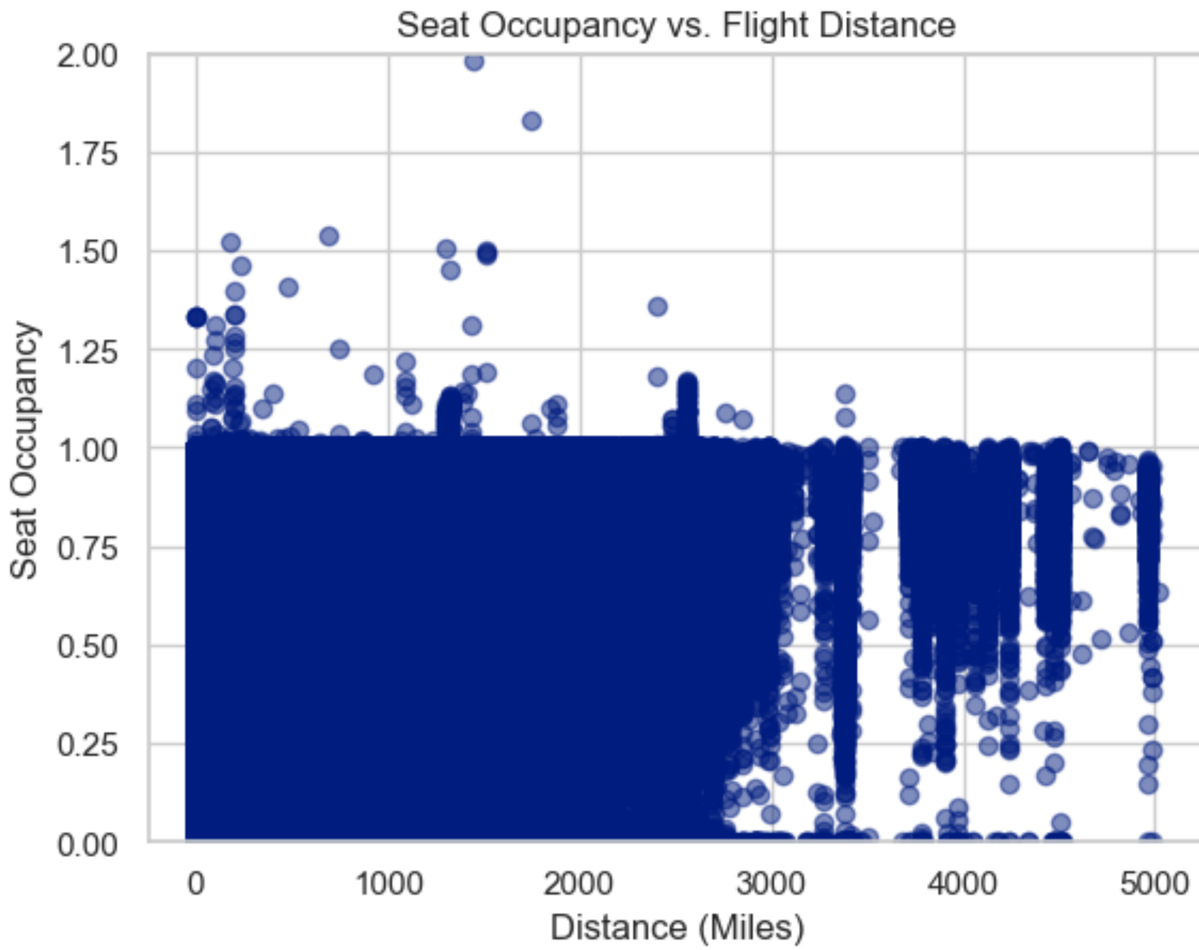
- As shown below, ***on an average, almost all flights took-off by atleast filling 55% of their seats capacity. We can also state that the seat occupancy is highest during summer vacation holidays as derived earlier.***

In [31]:
```
tasks.Seat_Occupancy_new_month()
```



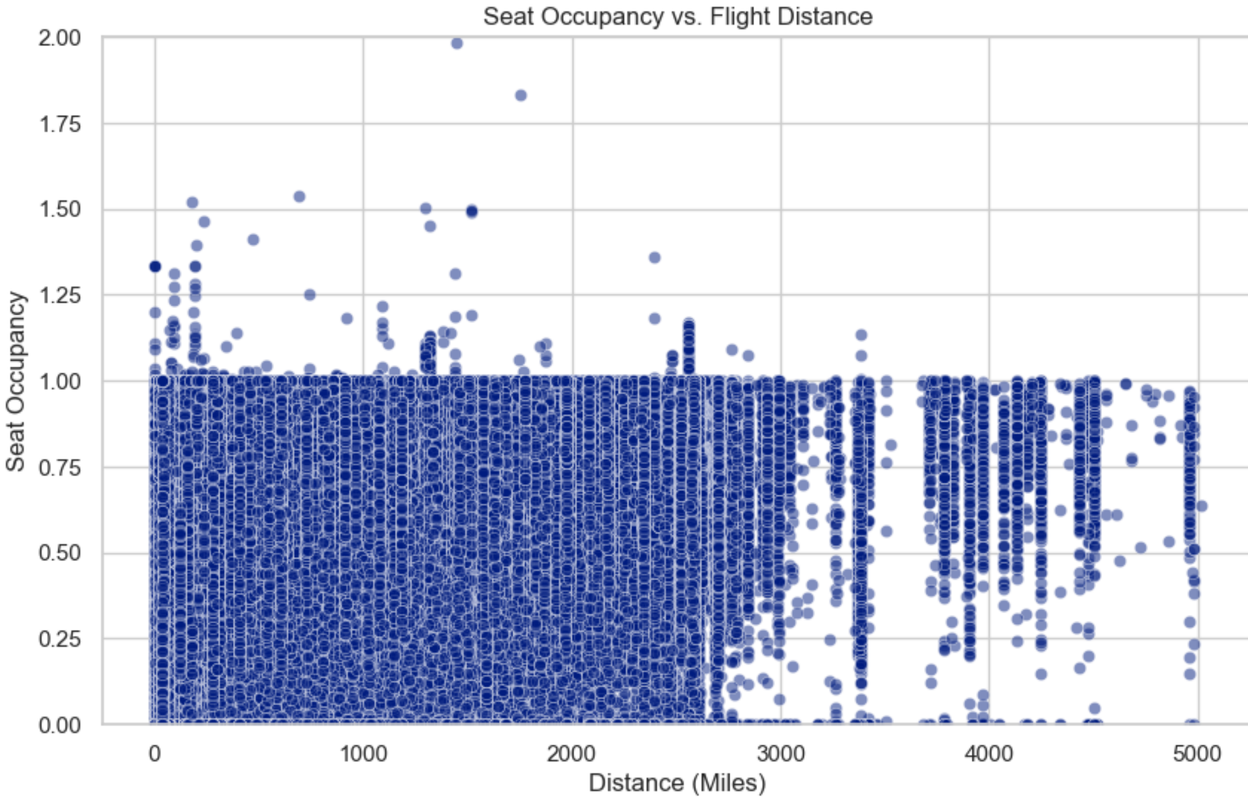
- As a supportive measure for the conclusion made from 2800 miles Distance Vs Passengers plot, let's plot a graph between Seat Occupancy and Distance over the entire dataset as shown below. Any value above Seat occupancy of 1 can be ignored as it is technically impossible to occure and this could be due to inadequacy in the data.

In [32]:
```
tasks.passengers_distance_matplotlib()
```



- With the help of seaborn plot, ***we can clearly see that seat occupancy is comparatively higher right after crossing a distance of 2800 miles and flights are taking off with filling atleast 30% of their capacity as the flight has to cross international borders and compensate for an increase in operating costs.***
- After 3800 miles distance, flights are not being flown with less seat occupancy rates which clearly supports the conclusion that has been derived earlier.

In [33]:
```
tasks.passengers_distance_seaborn()
```


Seat Occupancy vs. Flight Distance
Seat Occupancy
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1000
2000
3000
4000
5000
Distance (Miles)

Conclusion

Based on the ananlysis, it is concluded that there is a drastic drop in flight passengers from 2000 to 2002. The main reason for this could be the 9/11 attact that has occured in the year 2001 which affected US commercial airways to a great extent. Highest number of the passengers flew in Texas and California states and the reason behind this could be that those two states are highest populated and vast states in the Unites States after Alaska. Highest travelling occured during the summer vacation, holidays, events and festivals during june, july and august, where as the least because of unfavourable cold weather conditions for tourism during january and february. Flights were used the most for upto 1100 miles and there after there is a gradual drop in flights usage over increasing distance. There is a significant drop after 2700-2800 miles which can be clearly seen in seaborn plot comparatively. This conclusion stands on par with the longest line in america by Dr. Cliff Pickover which is 2802 miles as shown in his work. 2802miles is the longest distance connecting 2 extreme points in USA excluding Alaska and the increasing trend after nearly 3800 miles might be due to the crossing distance over Canada to Alaska. We can conclude that there is an increasing trend in seat occupancy from 1990 to 2009 except during 2001(due to 9/11 attack). On an average, almost all flights took-off by atleast filling 55% of their seats capacity. We can also state that the seat occupancy is highest during summer vacation holidays as derived earlier. The seat occupancy is comparatively higher right after crossing a distance of 2800 miles and flights are taking off with filling atleast 30% of their capacity as the flight has to cross international borders and compensate for an increase in operating costs.

References

- Chapter 7 in "Hands-On Data Analysis with Pandas" by Stefanie Molin
- https://numpy.org/doc/stable/reference/generated/numpy.ndarray.flatten.html
- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.copy.html
- https://academictorrents.com/details/a2ccf94bbb4af222bf8e69dad60a68a29f310d9a
- https://github.com/awesomedata/awesome-public-datasets

In [ ]: