

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 import os
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Dataset Loaded Successfully!

```

1 # Display dataset info (Column names, Data types, Missing values)
2 print("Dataset Info:")
3 df.info()
4

```

Dataset Info:

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 540 entries, 0 to 539
Data columns (total 5 columns):
 #   Column                                                                 Non-Null Count  Dtype
---  -
 0   Entity                                                                540 non-null   object
 1   Code                                                                  0 non-null     float64
 2   Year                                                                  540 non-null   int64
 3   City population (UN Urbanization Prospects, 2018)                  420 non-null   float64
 4   Projected city population (UN Urbanization Prospects, 2018)      120 non-null   float64
dtypes: float64(3), int64(1), object(1)
memory usage: 21.2+ KB

```

```

1 print(f"Dataset Shape: {df.shape}")
2 print(f"Total Elements in Dataset: {df.size}")
3

```

Dataset Shape: (540, 5)
Total Elements in Dataset: 2700

```

1 print("Column Names:", df.columns.tolist())
2

```

Column Names: ['Entity', 'Code', 'Year', 'City population (UN Urbanization Prospects, 2018)', 'Projected city population (UN Urbanization Prospects, 2018)']

```
1 df.index
```

RangeIndex(start=0, stop=540, step=1)

```
1 df.duplicated().sum()
```

np.int64(0)

```

1 n=int(input("enter the number of rows u want to view from the top"))
2 df.head(n)

```

enter the number of rows u want to view from the top3

	Entity	Code	Year	City population (UN Urbanization Prospects, 2018)	Projected city population (UN Urbanization Prospects, 2018)
0	Bangalore	NaN	1950	745999.0	NaN
1	Bangalore	NaN	1955	939396.0	NaN
2	Bangalore	NaN	1960	1165978.0	NaN

```
1 df.dtypes
```



```
Entity      object
Code        float64
Year         int64
City population (UN Urbanization Prospects, 2018)  float64
Projected city population (UN Urbanization Prospects, 2018)  float64
```

```
dtype: object
```

```
1 # Display the first five rows
2 df.head()
3
```



	Entity	Code	Year	City population (UN Urbanization Prospects, 2018)	Projected city population (UN Urbanization Prospects, 2018)
0	Bangalore	NaN	1950	745999.0	NaN
1	Bangalore	NaN	1955	939396.0	NaN
2	Bangalore	NaN	1960	1165978.0	NaN
3	Bangalore	NaN	1965	1377314.0	NaN
4	Bangalore	NaN	1970	1614756.0	NaN

```
1 df.describe()
```




	Code	Year	City population (UN Urbanization Prospects, 2018)	Projected city population (UN Urbanization Prospects, 2018)
count	0.0	540.00000	4.200000e+02	1.200000e+02
mean	NaN	1992.50000	8.449061e+06	1.939155e+07
std	NaN	25.96469	6.535627e+06	6.938304e+06
min	NaN	1950.00000	3.148000e+03	1.077049e+07
25%	NaN	1970.00000	3.394879e+06	1.436186e+07
50%	NaN	1992.50000	7.314234e+06	1.768104e+07
75%	NaN	2015.00000	1.181276e+07	2.281181e+07
max	NaN	2035.00000	3.725611e+07	4.334506e+07

```
1 # Rename columns for easier access
2 df.rename(columns={"City population (UN Urbanization Prospects, 2018)": "City Population"}, inplace=True)
3
4 # Drop rows where 'City Population' is missing
5 df_cleaned = df.dropna(subset=["City Population"])
6 print(f"Missing values handled. Remaining Rows: {df_cleaned.shape[0]}")
7
```



```
Missing values handled. Remaining Rows: 420
```

```
1 # Convert 'Year' column to integer
2 df_cleaned["Year"] = df_cleaned["Year"].astype(int)
3
4 # Display updated data types
5 df_cleaned.dtypes
6
```


 <ipython-input-16-96cc6d7f92f2>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df_cleaned["Year"] = df_cleaned["Year"].astype(int)


		0
	Entity	object
	Code	float64
	Year	int64
	City Population	float64
	Projected city population (UN Urbanization Prospects, 2018)	float64

 df_cleaned

```
1 # Remove duplicate rows
2 df_cleaned = df_cleaned.drop_duplicates()
3 print(f"Duplicate rows removed. Dataset now has {df_cleaned.shape[0]} rows.")
4
```


 Duplicate rows removed. Dataset now has 420 rows.

```
1 df_2020 = df.loc[df["Year"] == 2020]
2 df_2020.head()
3
```




	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
14	Bangalore	NaN	2020	NaN	12326532.0
32	Beijing	NaN	2020	NaN	20462610.0
50	Bogota	NaN	2020	NaN	10978360.0
68	Buenos Aires	NaN	2020	NaN	15153729.0
86	Cairo	NaN	2020	NaN	20900604.0

```
1 df_large_cities = df.loc[df["City Population"] > 10_000_000]
2 df_large_cities.head()
3
```



	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
13	Bangalore	NaN	2015	10141080.0	NaN
28	Beijing	NaN	2000	10285091.0	NaN
29	Beijing	NaN	2005	12991292.0	NaN
30	Beijing	NaN	2010	16441252.0	NaN
31	Beijing	NaN	2015	18421198.0	NaN

```
1 df_mumbai = df.loc[df["Entity"] == "Mumbai"]
2 df_mumbai.head()
3
```



	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
360	Mumbai	NaN	1950	3088811.0	NaN
361	Mumbai	NaN	1955	3726210.0	NaN
362	Mumbai	NaN	1960	4414904.0	NaN
363	Mumbai	NaN	1965	5314300.0	NaN
364	Mumbai	NaN	1970	6412876.0	NaN

```
1 df.iloc[:, [0, 2, 3]] # Retrieves the first, third, and fourth columns
2
```



	Entity	Year	City Population
0	Bangalore	1950	745999.0
1	Bangalore	1955	939396.0
2	Bangalore	1960	1165978.0
3	Bangalore	1965	1377314.0
4	Bangalore	1970	1614756.0
...
535	Tokyo	2015	37256109.0
536	Tokyo	2020	NaN
537	Tokyo	2025	NaN
538	Tokyo	2030	NaN
539	Tokyo	2035	NaN

540 rows × 3 columns

```
1 df.iloc[-5:, [0, 3]] # Retrieves the last 5 rows and only the first and fourth columns
```

2



	Entity	City Population
535	Tokyo	37256109.0
536	Tokyo	NaN
537	Tokyo	NaN
538	Tokyo	NaN
539	Tokyo	NaN

```
1 df.loc[100:105] # Retrieves rows from index 100 to 105 (inclusive)
```

2



	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
100	Chongqing	NaN	2000	7862976.0	NaN
101	Chongqing	NaN	2005	9454076.0	NaN
102	Chongqing	NaN	2010	11243667.0	NaN
103	Chongqing	NaN	2015	13372015.0	NaN
104	Chongqing	NaN	2020	NaN	15872179.0
105	Chongqing	NaN	2025	NaN	18171198.0

```
1 df["Entity"].value_counts()
```

```
2 # Returns a count of occurrences for each unique city
```

3



count

Entity

Bangalore	18
Beijing	18
Bogota	18
Buenos Aires	18
Cairo	18
Chongqing	18
Delhi	18
Dhaka	18
Guangzhou	18
Istanbul	18
Jakarta	18
Karachi	18
Kinshasa	18
Kolkata	18
Lagos	18
Lahore	18
Los Angeles	18
Manila	18
Mexico City	18
Moscow	18
Mumbai	18
New York	18
Osaka	18
Paris	18
Rio de Janeiro	18
Sao Paulo	18
Shanghai	18
Shenzhen	18
Tianjin	18
Tokyo	18

```
1 df.groupby("Year")["Entity"].nunique()
```

```
2 # Returns the count of unique cities for each year
```

```
3
```



Entity	
Year	
1950	30
1955	30
1960	30
1965	30
1970	30
1975	30
1980	30
1985	30
1990	30
1995	30
2000	30
2005	30
2010	30
2015	30
2020	30
2025	30
2030	30
2035	30


dtype: int64

```
1 df["Entity"].value_counts(normalize=True) * 100
2
3
```



	proportion
Entity	
Bangalore	3.333333
Beijing	3.333333
Bogota	3.333333
Buenos Aires	3.333333
Cairo	3.333333
Chongqing	3.333333
Delhi	3.333333
Dhaka	3.333333
Guangzhou	3.333333
Istanbul	3.333333
Jakarta	3.333333
Karachi	3.333333
Kinshasa	3.333333
Kolkata	3.333333
Lagos	3.333333
Lahore	3.333333
Los Angeles	3.333333
Manila	3.333333
Mexico City	3.333333
Moscow	3.333333
Mumbai	3.333333
New York	3.333333
Osaka	3.333333
Paris	3.333333
Rio de Janeiro	3.333333
Sao Paulo	3.333333
Shanghai	3.333333
Shenzhen	3.333333
Tianjin	3.333333
Tokyo	3.333333

```
1 df_sorted = df.sort_values(by=["Year", "City Population"], ascending=[False, True])
2 df_sorted.head()
3
```



	Entity	Code	Year	City Population	Projected city population (UN Urbanization Prospects, 2018)
17	Bangalore	NaN	2035	NaN	18065541.0
35	Beijing	NaN	2035	NaN	25365920.0
53	Bogota	NaN	2035	NaN	12753324.0
71	Buenos Aires	NaN	2035	NaN	17127741.0
89	Cairo	NaN	2035	NaN	28504351.0

```
1 # Identify and remove outliers using the Interquartile Range (IQR) method
2 q1 = df_cleaned["City Population"].quantile(0.25)
3 q3 = df_cleaned["City Population"].quantile(0.75)
4 iqr = q3 - q1
5
6 # Filter out outliers
7 df_filtered = df_cleaned[(df_cleaned["City Population"] >= q1 - 1.5 * iqr) &
8                           (df_cleaned["City Population"] <= q3 + 1.5 * iqr)]
```

```

9
10 print(f"Outliers removed. Remaining Rows: {df_filtered.shape[0]}")
11

```

Outliers removed. Remaining Rows: 410

```

1 # Select specific cities for visualization
2 selected_cities = ["New York", "London", "Tokyo", "Mumbai", "Bangalore"]
3 df_selected = df_filtered[df_filtered["Entity"].isin(selected_cities)]
4
5 print(f"Filtered Data for Selected Cities: {df_selected.shape[0]} rows")
6

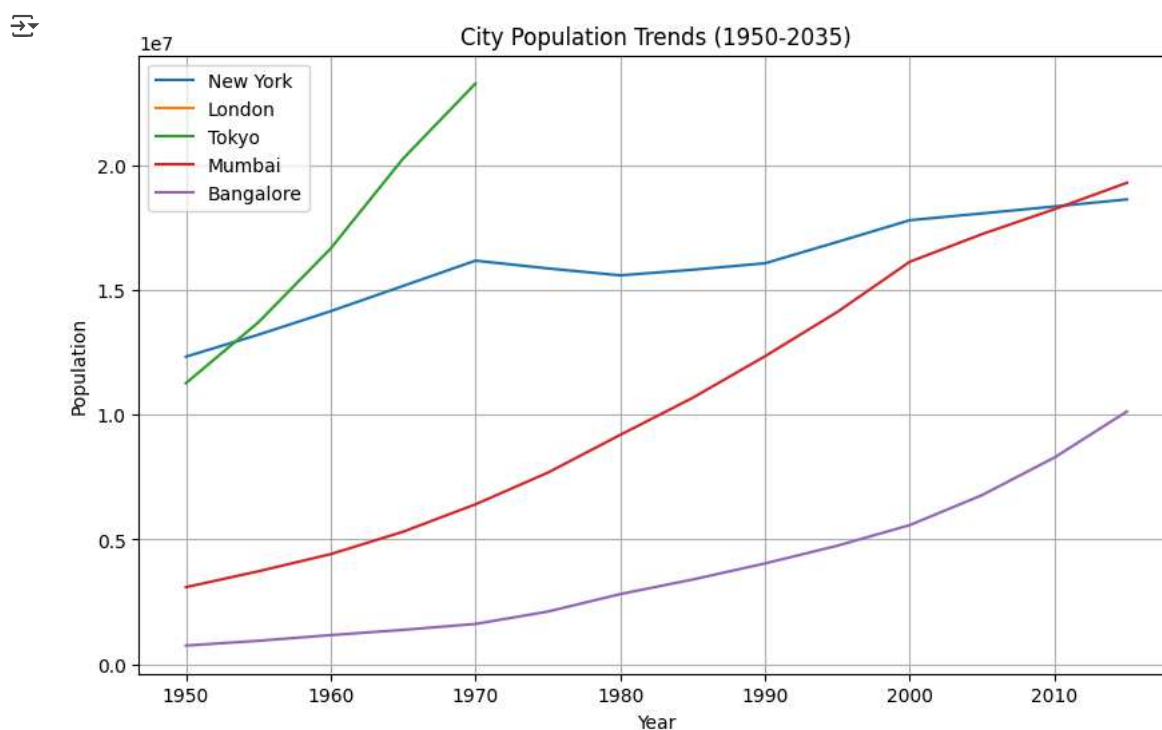
```

Filtered Data for Selected Cities: 47 rows

```

1 # Plot city population trends over time
2 plt.figure(figsize=(10, 6))
3
4 for city in selected_cities:
5     city_data = df_selected[df_selected["Entity"] == city]
6     plt.plot(city_data["Year"], city_data["City Population"], label=city)
7
8 # Customize the plot
9 plt.xlabel("Year")
10 plt.ylabel("Population")
11 plt.title("City Population Trends (1950-2035)")
12 plt.legend()
13 plt.grid(True)
14
15 # Show the plot
16 plt.show()
17

```



```

1 # Ensure valid data for the latest available year
2 latest_year = df["Year"].max()
3 df_latest = df[df["Year"] == latest_year].dropna(subset=["City Population"])
4
5 # If df_latest is still empty, use a previous year with valid data
6 if df_latest.empty:
7     valid_years = df.dropna(subset=["City Population"])["Year"].unique()
8     latest_year = max(valid_years) # Find the last available year with data
9     df_latest = df[df["Year"] == latest_year]
10
11 plt.figure(figsize=(12, 6))
12 top_cities = df_latest.nlargest(10, "City Population")

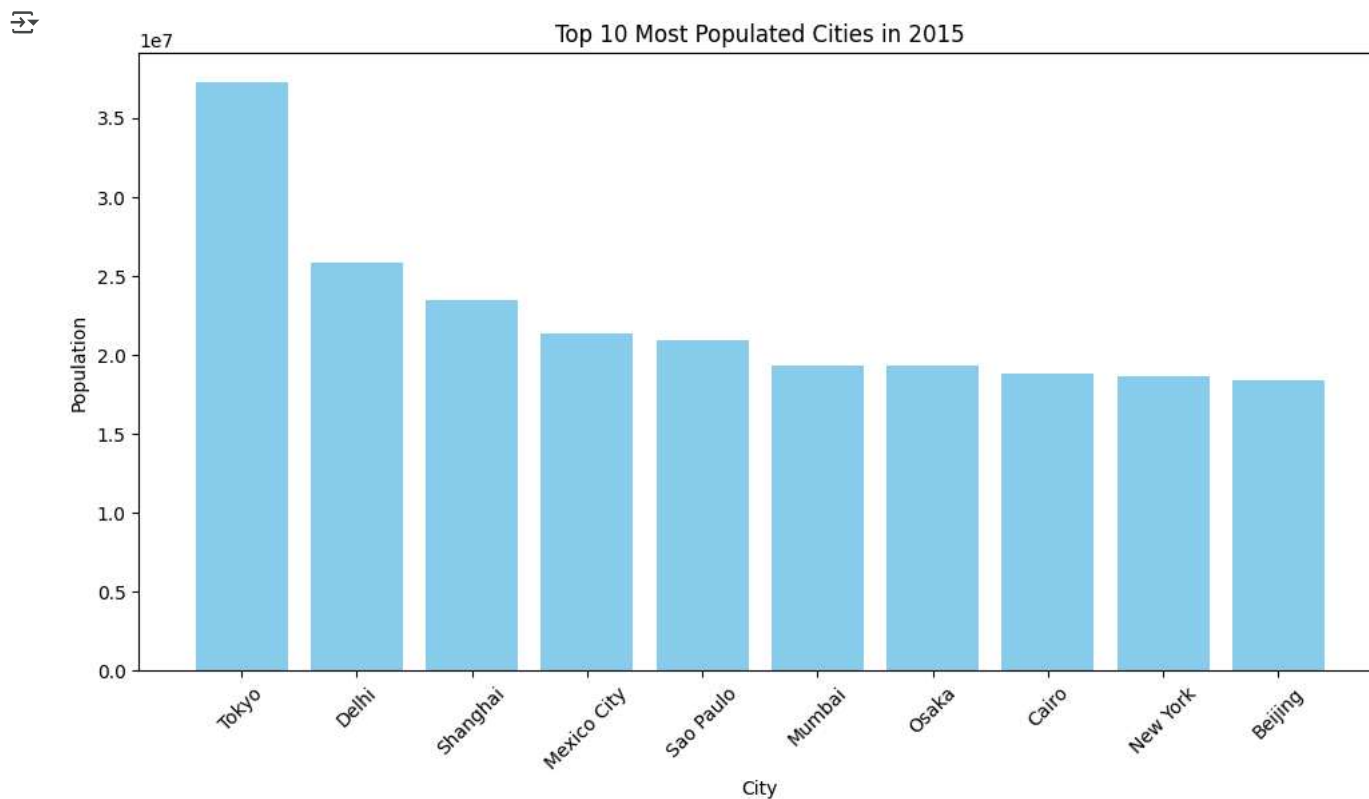
```



```

13 plt.bar(top_cities["Entity"], top_cities["City Population"], color='skyblue')
14 plt.xlabel("City")
15 plt.ylabel("Population")
16 plt.title(f"Top 10 Most Populated Cities in {latest_year}")
17 plt.xticks(rotation=45)
18 plt.show()

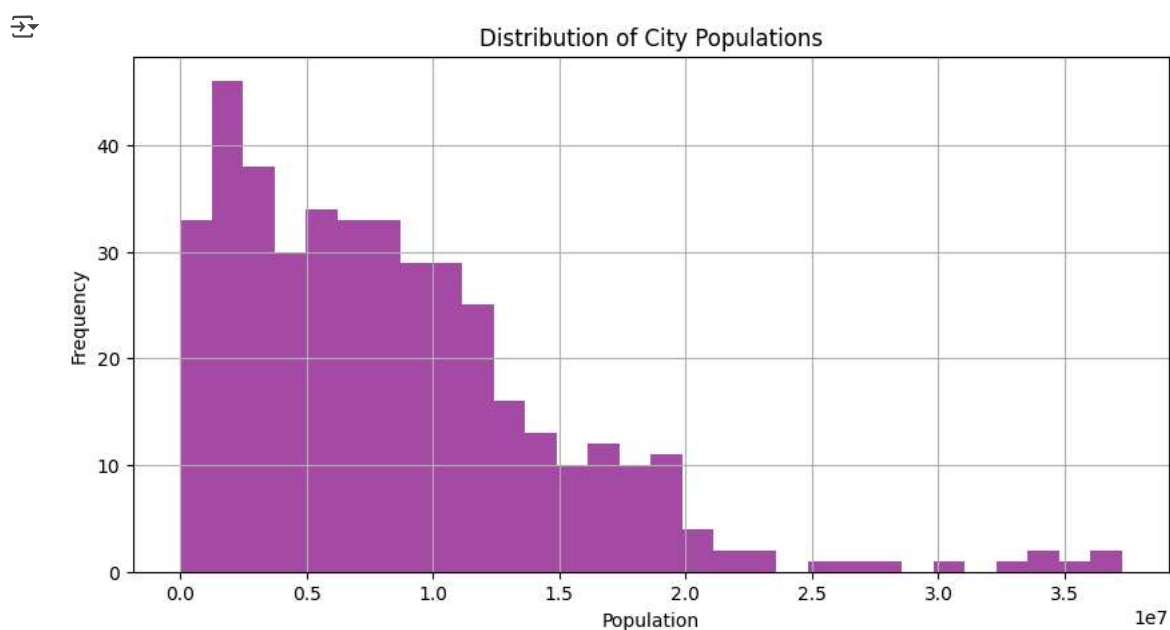
```



```

1 plt.figure(figsize=(10, 5))
2 plt.hist(df["City Population"], bins=30, color='purple', alpha=0.7)
3 plt.xlabel("Population")
4 plt.ylabel("Frequency")
5 plt.title("Distribution of City Populations")
6 plt.grid(True)
7 plt.show()

```

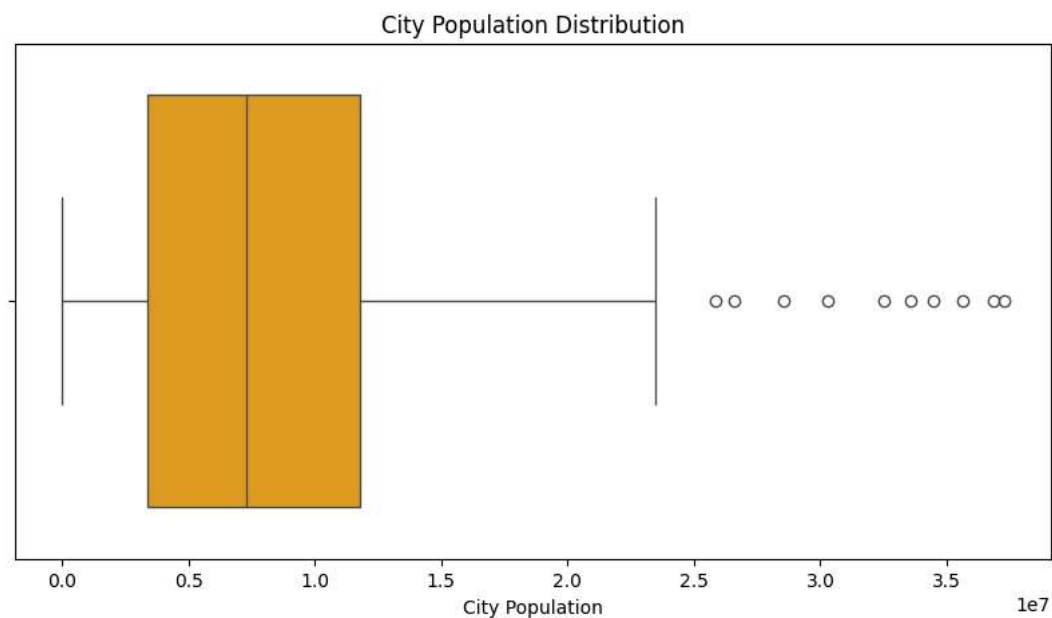


```

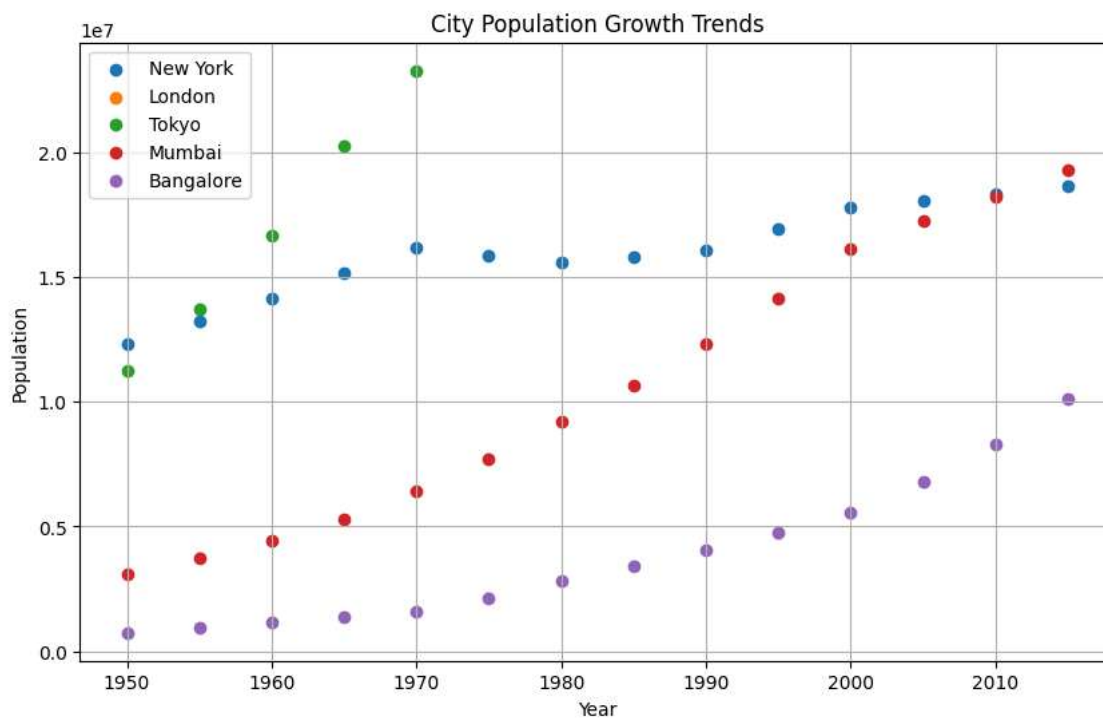
1 plt.figure(figsize=(10, 5))
2 sns.boxplot(x=df["City Population"], color='orange')

```

```
3 plt.title("City Population Distribution")
4 plt.show()
```



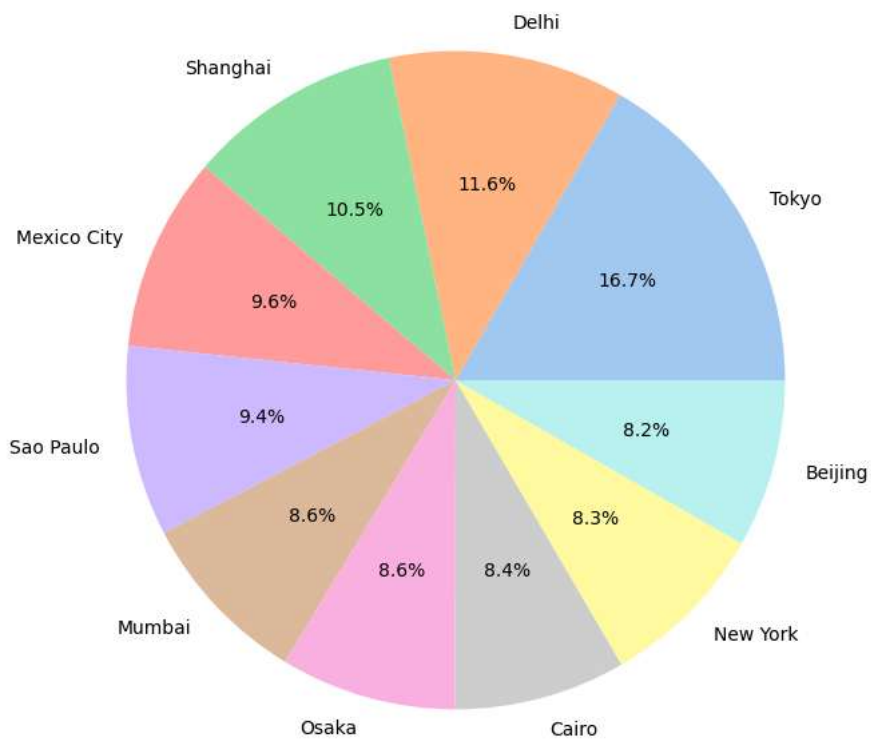
```
1 plt.figure(figsize=(10, 6))
2 for city in selected_cities:
3     city_data = df_selected[df_selected["Entity"] == city]
4     plt.scatter(city_data["Year"], city_data["City Population"], label=city)
5 plt.xlabel("Year")
6 plt.ylabel("Population")
7 plt.title("City Population Growth Trends")
8 plt.legend()
9 plt.grid()
10 plt.show()
```



```
1 plt.figure(figsize=(8, 8))
2 plt.pie(top_cities["City Population"], labels=top_cities["Entity"], autopct='%1.1f%%', colors=sns.color_palette("pastel"))
3 plt.title(f"Population Share of Top 10 Cities in {latest_year}")
4 plt.show()
```



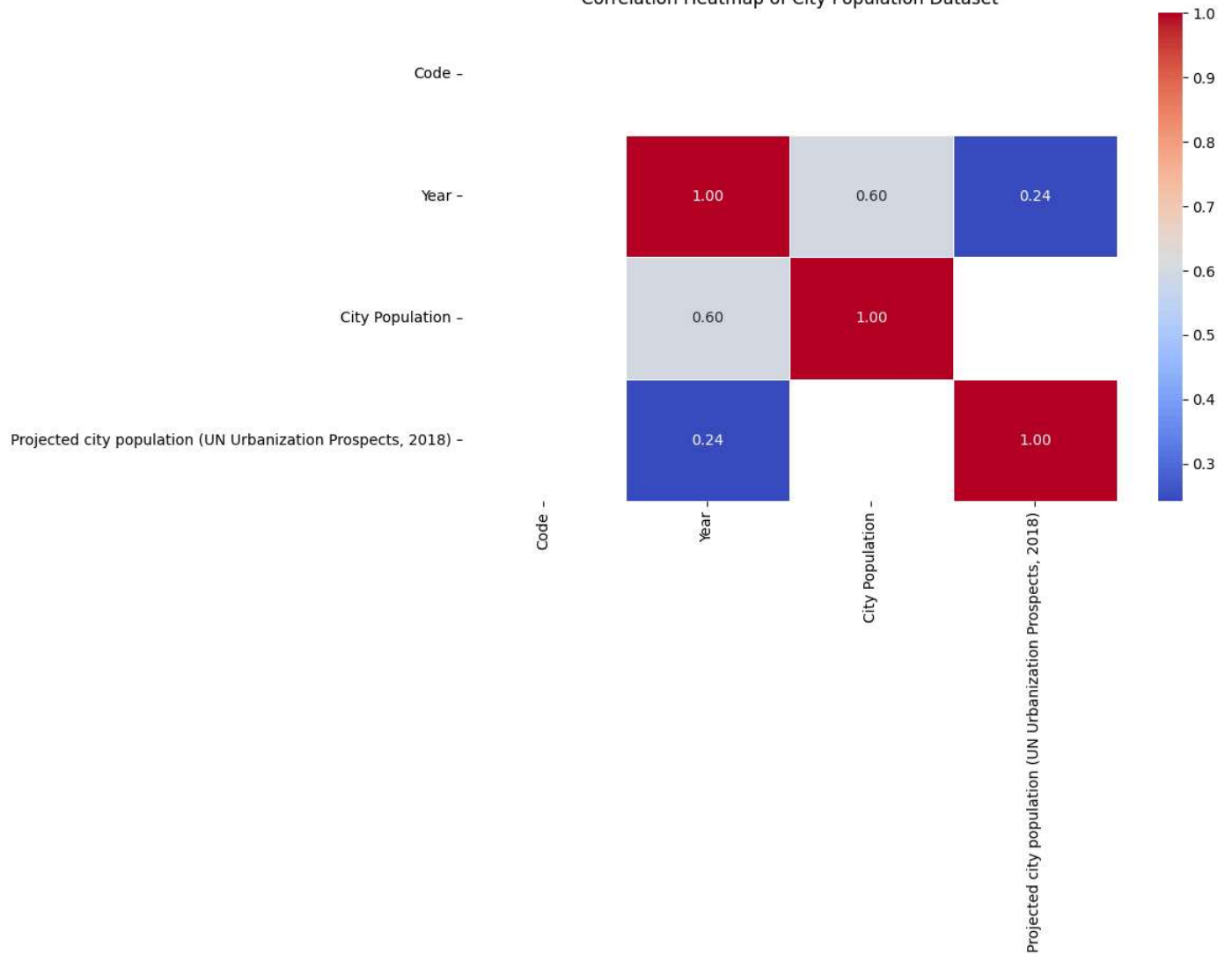
Population Share of Top 10 Cities in 2015



```
1 import seaborn as sns
2 import numpy as np
3
4
5 correlation_matrix = df.corr(numeric_only=True)
6
7 plt.figure(figsize=(10, 6))
8 sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
9 plt.title("Correlation Heatmap of City Population Dataset")
10 plt.show()
11
```



Correlation Heatmap of City Population Dataset



```

1 plt.figure(figsize=(10, 6))
2
3
4 scatter = plt.scatter(df_selected["Year"], df_selected["City Population"],
5                       c=df_selected["City Population"], cmap='coolwarm', alpha=0.75, edgecolors='k')
6
7 cbar = plt.colorbar(scatter)
8 cbar.set_label("Population")
9
10 plt.xlabel("Year")
11 plt.ylabel("Population")
12 plt.title("City Population Growth Trends (Color-Scaled by Population)")
13
14 plt.grid(True)
15 plt.show()

```

