

Anomaly Detection

In [47]:

```
import pandas as pd
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import LocalOutlierFactor
from sklearn.cluster import KMeans

# Load the dataset
df = pd.read_csv('bank-additional-full.csv', sep=';')

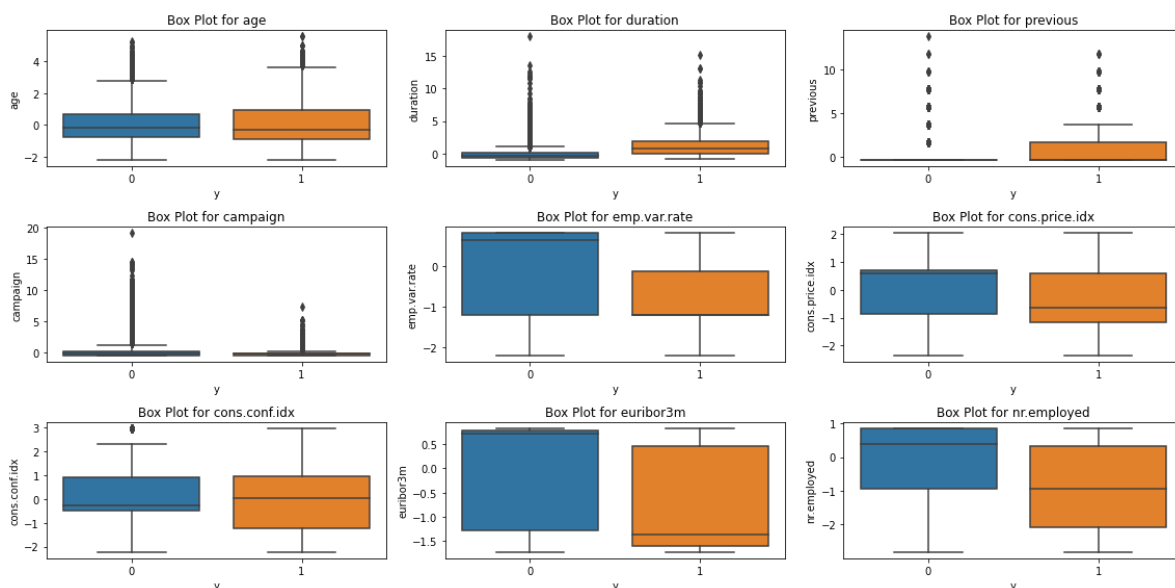
# Preprocess the data
data = pd.get_dummies(df, columns=['job', 'marital', 'education', 'housing', 'month',
data['y'] = df['y'].replace('no', '0')
data['y'] = data['y'].replace('yes', '1')
scaler = StandardScaler()
numerical_columns = ['age', 'duration', 'previous', 'campaign', 'emp.var.rate', 'cc
                    'cons.conf.idx', 'euribor3m', 'nr.employed']
data[numerical_columns] = scaler.fit_transform(df[numerical_columns])
```

In [16]:

Different Approaches for Anomaly Detection:

1- Box Plot for each variable

```
plt.figure(figsize=(16, 8))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(3, 3, i)
    sns.boxplot(x=data['y'], y=data[column])
    plt.title(f'Box Plot for {column}')
plt.tight_layout()
plt.show()
```

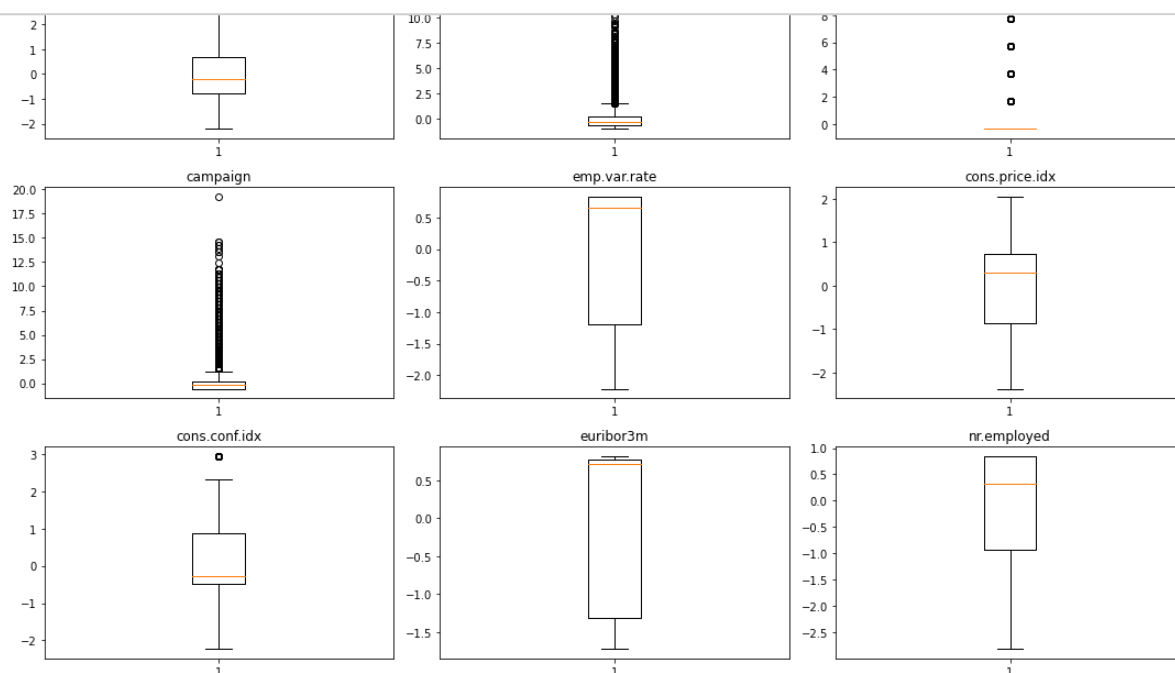


In [17]:

```
numerical columns for outlier detection
_columns = data[['age', 'duration', 'previous', 'campaign', 'emp.var.rate', 'cons.p

for each variable
e(figsize=(15, 10))
l in enumerate(numerical_columns.columns, 1):
ubplot(3, 3, i) # Use a 3x3 grid for the subplots
oxplot(numerical_columns[col])
itle(col)

_layout()
)
```



In [53]:

```
df[['age', 'duration', 'previous', 'campaign']].describe()
```

Out[53]:

	age	duration	previous	campaign
count	41188.00000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	0.172963	2.567593
std	10.42125	259.279249	0.494901	2.770014
min	17.00000	0.000000	0.000000	1.000000
25%	32.00000	102.000000	0.000000	1.000000
50%	38.00000	180.000000	0.000000	2.000000
75%	47.00000	319.000000	0.000000	3.000000
max	98.00000	4918.000000	7.000000	56.000000

With the plots and table above, I find that features such as age, balance, duration, campaign and previous all

In [28]:

```
#The average values of numerical variables for different Y values
print("The average values of numerical variables for different Y values: ")
df.pivot_table(['age', 'duration', 'previous', 'campaign'], ['y'], aggfunc='mean')
```

The average values of numerical variables for different Y values:

Out[28]:

	age	campaign	duration	previous
y				
no	39.911185	2.633085	220.844807	0.132374
yes	40.913147	2.051724	553.191164	0.492672

In [14]:

```
# Select numerical columns for outlier detection
numerical_columns = data[['age', 'duration', 'previous', 'campaign', 'emp.var.rate']]

# Calculate the first quartile (Q1) and third quartile (Q3)
Q1 = numerical_columns.quantile(0.25)
Q3 = numerical_columns.quantile(0.75)

# Calculate the Interquartile Range (IQR)
IQR = Q3 - Q1

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

# Identify outliers
outliers = ((numerical_columns < lower_bound) | (numerical_columns > upper_bound))

# Display the number of outliers
print("Number of outliers:", outliers.sum())

# Display the indices of the outliers
print("Indices of outliers:", data.index[outliers])
```

Number of outliers: 10828

Indices of outliers: Int64Index([37, 57, 61, 75, 83, 88, 111, 131, 164, 169, ..., 41164, 41166, 41170, 41173, 41174, 41175, 41178, 41182, 41183, 41187], dtype='int64', length=10828)

In [33]:

```
# Display all observations that have outliers  
print("Observations with outliers:")  
print(data[outliers])
```

Observations with outliers:

	age	default	loan	contact	duration	campaign	pdays
previous \							
37	1.149199	no	no	telephone	5.429405	-0.565922	999 -
0.349494							
57	0.477486	unknown	no	telephone	2.039197	-0.565922	999 -
0.349494							
61	1.053240	no	no	telephone	2.135619	-0.565922	999 -
0.349494							
75	0.093650	unknown	no	telephone	5.078428	-0.565922	999 -
0.349494							
83	0.861322	unknown	no	telephone	3.022704	-0.565922	999 -
0.349494							
...
...							
41175	-0.578062	no	no	cellular	-0.301937	-0.565922	999
3.691766							
41178	2.108788	no	no	cellular	0.866702	-0.204909	6
5.712397							
41182	-1.057857	no	no	cellular	-0.564206	-0.565922	9
1.671136							
41183	3.164336	no	no	cellular	0.292025	-0.565922	999 -
0.349494							
41187	3.260295	no	no	cellular	-0.074380	0.156105	999
1.671136							

	poutcome	emp.var.rate	...	month_mar	month_may	month_no
v \						
37	nonexistent	0.648092	...	0	1	
0						
57	nonexistent	0.648092	...	0	1	
0						
61	nonexistent	0.648092	...	0	1	
0						
75	nonexistent	0.648092	...	0	1	
0						
83	nonexistent	0.648092	...	0	1	
0						
...	
...						
41175	failure	-0.752343	...	0	0	
1						
41178	success	-0.752343	...	0	0	
1						
41182	success	-0.752343	...	0	0	
1						
41183	nonexistent	-0.752343	...	0	0	
1						
41187	failure	-0.752343	...	0	0	
1						

	month_oct	month_sep	day_of_week_fri	day_of_week_mon	day_of_
week_thu \					
37	0	0	0	1	
0					
57	0	0	0	1	
0					
61	0	0	0	1	
0					
75	0	0	0	1	

```

0
83          0          0          0          1
0
...          ...          ...          ...          ...
...
41175        0          0          0          0
1
41178        0          0          0          0
1
41182        0          0          1          0
0
41183        0          0          1          0
0
41187        0          0          1          0
0

```

```

          day_of_week_tue  day_of_week_wed
37                0          0
57                0          0
61                0          0
75                0          0
83                0          0
...                ...          ...
41175              0          0
41178              0          0
41182              0          0
41183              0          0
41187              0          0

```

[10828 rows x 57 columns]

In [49]:

```

# 2- Distance-based Approach: KNN Outlier Detection
features_distance = data[numerical_cols]
lof = LocalOutlierFactor(n_neighbors=5, contamination=0.1)
outlier_scores_distance = lof.fit_predict(features_distance)
outliers_distance = features_distance[outlier_scores_distance == -1]
print("Number of Distance-based outliers:", len(outliers_distance))
outlier_indices_distance = data.index[outlier_scores_distance == -1]
print("Indices of Distance-based outliers:", outlier_indices_distance)

```

```

Number of Distance-based outliers: 4119
Indices of Distance-based outliers: Int64Index([ 12,    62,    80,
 84,    97,   127,   139,   140,   158,
      162,
      ...
 41152, 41153, 41154, 41156, 41163, 41164, 41165, 41178, 4
1186,
      41187],
      dtype='int64', length=4119)

```

In [50]:

```
# 2- Distance-based Approach: KNN Outlier Detection
features_distance = data[numerical_cols]
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.1)
outlier_scores_distance = lof.fit_predict(features_distance)
outliers_distance = features_distance[outlier_scores_distance == -1]
print("Number of Distance-based outliers:", len(outliers_distance))
outlier_indices_distance = data.index[outlier_scores_distance == -1]
print("Indices of Distance-based outliers:", outlier_indices_distance)
```

```
Number of Distance-based outliers: 4119
Indices of Distance-based outliers: Int64Index([    8,    26,    31,
84,   160,   163,   192,   228,   278,
      288,
      ...
41157, 41160, 41161, 41164, 41166, 41174, 41177, 41184, 4
1186,
      41187],
dtype='int64', length=4119)
```

In [51]:

```
# 2- Distance-based Approach: KNN Outlier Detection
features_distance = data[numerical_cols]
lof = LocalOutlierFactor(n_neighbors=100, contamination=0.1)
outlier_scores_distance = lof.fit_predict(features_distance)
outliers_distance = features_distance[outlier_scores_distance == -1]
print("Number of Distance-based outliers:", len(outliers_distance))
outlier_indices_distance = data.index[outlier_scores_distance == -1]
print("Indices of Distance-based outliers:", outlier_indices_distance)
```

```
Number of Distance-based outliers: 4119
Indices of Distance-based outliers: Int64Index([    9,    20,    84,
105,   110,   111,   192,   244,   288,
      355,
      ...
41123, 41124, 41126, 41136, 41144, 41153, 41159, 41164, 4
1174,
      41183],
dtype='int64', length=4119)
```

In [35]:

```
# Display all observations that have Distance-based outliers
print("Observations with Distance-based outliers:")
print(data.loc[outlier_indices_distance])
```

Observations with Distance-based outliers:

	previous	age	default	loan	contact	duration	campaign	pdays	
8	-1.537652		no	no	telephone	0.469442	-0.565922	999	-
0.349494									
26	1.820911		no	no	telephone	-0.637486	-0.565922	999	-
0.349494									
31	1.820911		no	no	telephone	0.492583	-0.565922	999	-
0.349494									
84	-0.194227		no	yes	telephone	-0.919040	-0.565922	999	-
0.349494									
160	-1.537652		no	no	telephone	-0.359790	-0.204909	999	-
0.349494									
...	
...									
41174	2.108788		no	no	cellular	-0.193944	-0.565922	1	1
1.774288									
41177	1.628993		no	no	cellular	-0.517923	1.239145	999	-
0.349494									
...	

In [7]:

```
# 3- Density-based Approach: DBSCAN
features_density = data[numerical_cols]
dbscan = DBSCAN(eps=0.5, min_samples=5)
labels_density = dbscan.fit_predict(features_density)
outliers_density = features_density[labels_density == -1]
print("Number of Density-based outliers:", len(outliers_density))
```

Number of Density-based outliers: 2670

In [41]:

```
# Identify outliers based on DBSCAN labels
outliers_density_indices = data.index[labels_density == -1].tolist()
outliers_density = data.loc[outliers_density_indices]

print("Number of Density-based outliers:", len(outliers_density_indices))

# Display the indices of the Density-based outliers
print("Indices of Density-based outliers:", outliers_density_indices)

# Display all observations that have Density-based outliers
if not outliers_density.empty:
    print("Observations with Density-based outliers:")
    print(outliers_density)
else:
    print("No observations with Density-based outliers.")
```

Number of Density-based outliers: 2670

Indices of Density-based outliers: [37, 164, 590, 943, 1114, 1396, 1689, 1791, 1809, 1839, 1853, 1980, 2031, 2105, 2313, 2330, 2610, 2970, 3127, 3219, 3370, 3413, 3418, 3427, 3438, 3484, 3514, 3532, 3539, 3652, 3671, 3770, 3772, 3773, 3774, 3785, 3809, 3817, 3854, 3868, 3872, 3892, 4039, 4045, 4056, 4107, 4114, 4139, 4140, 4152, 4164, 4168, 4176, 4213, 4221, 4264, 4353, 4410, 4456, 4575, 4650, 4847, 4880, 4897, 4902, 4942, 4978, 5017, 5043, 5073, 5112, 5304, 5337, 5384, 5386, 5415, 5476, 5530, 5550, 5564, 5699, 5784, 5791, 5894, 5948, 6100, 6203, 6280, 6365, 6394, 6531, 6534, 6619, 6738, 6778, 6860, 7155, 7251, 7277, 7297, 7302, 7390, 7542, 7544, 7613, 7629, 7700, 7714, 7719, 7727, 7753, 7841, 7945, 8014, 8016, 8097, 8099, 8101, 8102, 8125, 8130, 8222, 8233, 8246, 8301, 8339, 8346, 8362, 8379, 8417, 8435, 8437, 8455, 8471, 8489, 8528, 8529, 8617, 8637, 8640, 8643, 8661, 8712, 8740, 8765, 8952, 9013, 9048, 9057, 9072, 9095, 9149, 9160, 9196, 9205, 9251, 9258, 9276, 9346, 9432, 9487, 9524, 9759, 9785, 9811, 9937, 9951, 9969, 9974, 9987, 9988, 9991, 9999, 10016, 10025, 10037, 10061, 10114, 10124, 10157, 10158, 10160, 10162, 10184, 10224, 10233, 10239, 10254, 10299, 10310, 10337, 10354, 10431, 10455, 10456, 10466, 10478, 10488, 10500, 10550, 10555, 10558, 10559, 10560, 10561, 10562, 10563, 10564, 10565, 10566, 10567, 10568, 10569, 10570, 10571, 10572, 10573, 10574, 10575, 10576, 10577, 10578, 10579, 10580, 10581, 10582, 10583, 10584, 10585, 10586, 10587, 10588, 10589, 10590, 10591, 10592, 10593, 10594, 10595, 10596, 10597, 10598, 10599, 10600, 10601, 10602, 10603, 10604, 10605, 10606, 10607, 10608, 10609, 10610, 10611, 10612, 10613, 10614, 10615, 10616, 10617, 10618, 10619, 10620, 10621, 10622, 10623, 10624, 10625, 10626, 10627, 10628, 10629, 10630, 10631, 10632, 10633, 10634, 10635, 10636, 10637, 10638, 10639, 10640, 10641, 10642, 10643, 10644, 10645, 10646, 10647, 10648, 10649, 10650, 10651, 10652, 10653, 10654, 10655, 10656, 10657, 10658, 10659, 10660, 10661, 10662, 10663, 10664, 10665, 10666, 10667, 10668, 10669, 10670, 10671, 10672, 10673, 10674, 10675, 10676, 10677, 10678, 10679, 10680, 10681, 10682, 10683, 10684, 10685, 10686, 10687, 10688, 10689, 10690, 10691, 10692, 10693, 10694, 10695, 10696, 10697, 10698, 10699, 10700, 10701, 10702, 10703, 10704, 10705, 10706, 10707, 10708, 10709, 10710, 10711, 10712, 10713, 10714, 10715, 10716, 10717, 10718, 10719, 10720, 10721, 10722, 10723, 10724, 10725, 10726, 10727, 10728, 10729, 10730, 10731, 10732, 10733, 10734, 10735, 10736, 10737, 10738, 10739, 10740, 10741, 10742, 10743, 10744, 10745, 10746, 10747, 10748, 10749, 10750, 10751, 10752, 10753, 10754, 10755, 10756, 10757, 10758, 10759, 10760, 10761, 10762, 10763, 10764, 10765, 10766, 10767, 10768, 10769, 10770, 10771, 10772, 10773, 10774, 10775, 10776, 10777, 10778, 10779, 10780, 10781, 10782, 10783, 10784, 10785, 10786, 10787, 10788, 10789, 10790, 10791, 10792, 10793, 10794, 10795, 10796, 10797, 10798, 10799, 10800, 10801, 10802, 10803, 10804, 10805, 10806, 10807, 10808, 10809, 10810, 10811, 10812, 10813, 10814, 10815, 10816, 10817, 10818, 10819, 10820, 10821, 10822, 10823, 10824, 10825, 10826, 10827, 10828, 10829, 10830, 10831, 10832, 10833, 10834, 10835, 10836, 10837, 10838, 10839, 10840, 10841, 10842, 10843, 10844, 10845, 10846, 10847, 10848, 10849, 10850, 10851, 10852, 10853, 10854, 10855, 10856, 10857, 10858, 10859, 10860, 10861, 10862, 10863, 10864, 10865, 10866, 10867, 10868, 10869, 10870, 10871, 10872, 10873, 10874, 10875, 10876, 10877, 10878, 10879, 10880, 10881, 10882, 10883, 10884, 10885, 10886, 10887, 10888, 10889, 10890, 10891, 10892, 10893, 10894, 10895, 10896, 10897, 10898, 10899, 10900, 10901, 10902, 10903, 10904, 10905, 10906, 10907, 10908, 10909, 10910, 10911, 10912, 10913, 10914, 10915, 10916, 10917, 10918, 10919, 10920, 10921, 10922, 10923, 10924, 10925, 10926, 10927, 10928, 10929, 10930, 10931, 10932, 10933, 10934, 10935, 10936, 10937, 10938, 10939, 10940, 10941, 10942, 10943, 10944, 10945, 10946, 10947, 10948, 10949, 10950, 10951, 10952, 10953, 10954, 10955, 10956, 10957, 10958, 10959, 10960, 10961, 10962, 10963, 10964, 10965, 10966, 10967, 10968, 10969, 10970, 10971, 10972, 10973, 10974, 10975, 10976, 10977, 10978, 10979, 10980, 10981, 10982, 10983, 10984, 10985, 10986, 10987, 10988, 10989, 10990, 10991, 10992, 10993, 10994, 10995, 10996, 10997, 10998, 10999, 11000, 11001, 11002, 11003, 11004, 11005, 11006, 11007, 11008, 11009, 11010, 11011, 11012, 11013, 11014, 11015, 11016, 11017, 11018, 11019, 11020, 11021, 11022, 11023, 11024, 11025, 11026, 11027, 11028, 11029, 11030, 11031, 11032, 11033, 11034, 11035, 11036, 11037, 11038, 11039, 11040, 11041, 11042, 11043, 11044, 11045, 11046, 11047, 11048, 11049, 11050, 11051, 11052, 11053, 11054, 11055, 11056, 11057, 11058, 11059, 11060, 11061, 11062, 11063, 11064, 11065, 11066, 11067, 11068, 11069, 11070, 11071, 11072, 11073, 11074, 11075, 11076, 11077, 11078, 11079, 11080, 11081, 11082, 11083, 11084, 11085, 11086, 11087, 11088, 11089, 11090, 11091, 11092, 11093, 11094, 11095, 11096, 11097, 11098, 11099, 11100, 11101, 11102, 11103, 11104, 11105, 11106, 11107, 11108, 11109, 11110, 11111, 11112, 11113, 11114, 11115, 11116, 11117, 11118, 11119, 11120, 11121, 11122, 11123, 11124, 11125, 11126, 11127, 11128, 11129, 11130, 11131, 11132, 11133, 11134, 11135, 11136, 11137, 11138, 11139, 11140, 11141, 11142, 11143, 11144, 11145, 11146, 11147, 11148, 11149, 11150, 11151, 11152, 11153, 11154, 11155, 11156, 11157, 11158, 11159, 11160, 11161, 11162, 11163, 11164, 11165, 11166, 11167, 11168, 11169, 11170, 11171, 11172, 11173, 11174, 11175, 11176, 11177, 11178, 11179, 11180, 11181, 11182, 11183, 11184, 11185, 11186, 11187, 11188, 11189, 11190, 11191, 11192, 11193, 11194, 11195, 11196, 11197, 11198, 11199, 11200, 11201, 11202, 11203, 11204, 11205, 11206, 11207, 11208, 11209, 11210, 11211, 11212, 11213, 11214, 11215, 11216, 11217, 11218, 11219, 11220, 11221, 11222, 11223, 11224, 11225, 11226, 11227, 11228, 11229, 11230, 11231, 11232, 11233, 11234, 11235, 11236, 11237, 11238, 11239, 11240, 11241, 11242, 11243, 11244, 11245, 11246, 11247, 11248, 11249, 11250, 11251, 11252, 11253, 11254, 11255, 11256, 11257, 11258, 11259, 11260, 11261, 11262, 11263, 11264, 11265, 11266, 11267, 11268, 11269, 11270, 11271, 11272, 11273, 11274, 11275, 11276, 11277, 11278, 11279, 11280, 11281, 11282, 11283, 11284, 11285, 11286, 11287, 11288, 11289, 11290, 11291, 11292, 11293, 11294, 11295, 11296, 11297, 11298, 11299, 11300, 11301, 11302, 11303, 11304, 11305, 11306, 11307, 11308, 11309, 11310, 11311, 11312, 11313, 11314, 11315, 11316, 11317, 11318, 11319, 11320, 11321, 11322, 11323, 11324, 11325, 11326, 11327, 11328, 11329, 11330, 11331, 11332, 11333, 11334, 11335, 11336, 11337, 11338, 11339, 11340, 11341, 11342, 11343, 11344, 11345, 11346, 11347, 11348, 11349, 11350, 11351, 11352, 11353, 11354, 11355, 11356, 11357, 11358, 11359, 11360, 11361, 11362, 11363, 11364, 11365, 11366, 11367, 11368, 11369, 11370, 11371, 11372, 11373, 11374, 11375, 11376, 11377, 11378, 11379, 11380, 11381, 11382, 11383, 11384, 11385, 11386, 11387, 11388, 11389, 11390, 11391, 11392, 11393, 11394, 11395, 11396, 11397, 11398, 11399, 11400, 11401, 11402, 11403, 11404, 11405, 11406, 11407, 11408, 11409, 11410, 11411, 11412, 11413, 11414, 11415, 11416, 11417, 11418, 11419, 11420, 11421, 11422, 11423, 11424, 11425, 11426, 11427, 11428, 11429, 11430, 11431, 11432, 11433, 11434, 11435, 11436, 11437, 11438, 11439, 11440, 11441, 11442, 11443, 11444, 11445, 11446, 11447, 11448, 11449, 11450, 11451, 11452, 11453, 11454, 11455, 11456, 11457, 11458, 11459, 11460, 11461, 11462, 11463, 11464, 11465, 11466, 11467, 11468, 11469, 11470, 11471, 11472, 11473, 11474, 11475, 11476, 11477, 11478, 11479, 11480, 11481, 11482, 11483, 11484, 11485, 11486, 11487, 11488, 11489, 11490, 11491, 11492, 11493, 11494, 11495, 11496, 11497, 11498, 11499, 11500, 11501, 11502, 11503, 11504, 11505, 11506, 11507, 11508, 11509, 11510, 11511, 11512, 11513, 11514, 11515, 11516, 11517, 11518, 11519, 11520, 11521, 11522, 11523, 11524, 11525, 11526, 11527, 11528, 11529, 11530, 11531, 11532, 11533, 11534, 11535, 11536, 11537, 11538, 11539, 11540, 11541, 11542, 11543, 11544, 11545, 11546, 11547, 11548, 11549, 11550, 11551, 11552, 11553, 11554, 11555, 11556, 11557, 11558, 11559, 11560, 11561, 11562, 11563, 11564, 11565, 11566, 11567, 11568, 11569, 11570, 11571, 11572, 11573, 11574, 11575, 11576, 11577, 11578, 11579, 11580, 11581, 11582, 11583, 11584, 11585, 11586, 11587, 11588, 11589, 11590, 11591, 11592, 11593, 11594, 11595, 11596, 11597, 11598, 11599, 11600, 11601, 11602, 11603, 11604, 11605, 11606, 11607, 11608, 11609, 11610, 11611, 11612, 11613, 11614, 11615, 11616, 11617, 11618, 11619, 11620, 11621, 11622, 11623, 11624, 11625, 11626, 11627, 11628, 11629, 11630, 11631, 11632, 11633, 11634, 11635, 11636, 11637, 11638, 11639, 11640, 11641, 11642, 11643, 11644, 11645, 11646, 11647, 11648, 11649, 11650, 11651, 11652, 11653, 11654, 11655, 11656, 11657, 11658, 11659, 11660, 11661, 11662, 11663, 11664, 11665, 11666, 11667, 11668, 11669, 11670, 11671, 11672, 11673, 11674, 11675, 11676, 11677, 11678, 11679, 11680, 11681, 11682, 11683, 11684, 11685, 11686, 11687, 11688, 11689, 11690, 11691, 11692, 11693, 11694, 11695, 11696, 11697, 11698, 11699, 11700, 11701, 11702, 11703, 11704, 11705, 11706, 11707, 11708, 11709, 11710, 11711, 11712, 11713, 11714, 11715, 11716, 11717, 11718, 11719, 11720, 11721, 11722, 11723, 11724, 11725, 11726, 11727, 11728, 11729, 11730, 11731, 11732, 11733, 11734, 11735, 11736, 11737, 11738, 11739, 11740, 11741, 11742, 11743, 11744, 11745, 11746, 11747, 11748, 11749, 11750, 11751, 11752, 11753, 11754, 11755, 11756, 11757, 11758, 11759, 11760, 11761, 11762, 11763, 11764, 11765, 11766, 11767, 11768, 11769, 11770, 11771, 11772, 11773, 11774, 11775, 11776, 11777, 11778, 11779, 11780, 11781, 11782, 11783, 11784, 11785, 11786, 11787, 11788, 11789, 11790, 11791, 11792, 11793, 11794, 11795, 11796, 11797, 11798, 11799, 11800, 11801, 11802, 11803, 11804, 11805, 11806, 11807, 11808, 11809, 11810, 11811, 11812, 11813, 11814, 11815, 11816, 11817, 11818, 11819, 11820, 11821, 11822, 11823, 11824, 11825, 11826, 11827, 11828, 11829, 11830, 11831, 11832, 11833, 11834, 11835, 11836, 11837, 11838, 11839, 11840, 11841, 11842, 11843, 11844, 11845, 11846, 11847, 11848, 11849, 11850, 11851, 11852, 11853, 11854, 11855, 11856, 11857, 11858, 11859, 11860, 11861, 11862, 11863, 11864, 11865, 11866, 11867, 11868, 11869, 11870, 11871, 11872, 11873, 11874, 11875, 11876, 11877, 11878, 11879, 11880, 11881, 11882, 11883, 11884, 11885, 11886, 11887, 11888, 11889, 11890, 11891, 11892, 11893, 11894, 11895, 11896, 11897, 11898, 11899, 11900, 11901, 11902, 11903, 11904, 11905, 11906, 11907, 11908, 11909, 11910, 11911, 11912, 11913, 11914, 11915, 11916, 11917, 11918, 11919, 11920, 11921, 11922, 11923, 11924, 11925, 11926, 11927, 11928, 11929, 11930, 11931, 11932, 11933, 11934, 11935, 11936, 11937, 11938, 11939, 11940, 11941, 11942, 11943, 11944, 11945, 11946, 11947, 11948, 11949, 11950, 11951, 11952, 11953, 11954, 11955, 11956, 11957, 11958, 11959, 11960, 11961, 11962, 11963, 11964, 11965, 11966, 11967, 11968, 11969, 11970, 11971, 11972, 11973, 11974, 11975, 11976, 11977, 11978, 11979, 11980, 11981, 11982, 11983, 11984, 11985, 11986, 11987, 11988, 11989, 11990, 11991, 11992, 11993, 11994, 11995, 11996, 11997, 11998, 11999, 12000, 12001, 12002, 12003, 12004, 12005, 12006, 12007, 12008, 12009, 12010, 12011, 12012, 12013, 12014, 12015, 12016, 12017, 12018, 12019, 12020, 12021, 12022, 12023, 12024, 12025, 12026, 12027, 12028, 12029, 12030, 12031, 12032, 12033, 12034, 12035, 12036, 12037, 12038, 12039, 12040, 12041, 12042, 12043, 12044, 12045, 12046, 12047, 12048, 12049, 12050, 12051, 12052, 12053, 12054, 12055, 12056, 12057, 12058, 12059, 12060, 12061, 12062, 12063, 12064, 12065, 12066, 12067, 12068, 12069, 12070, 12071, 12072, 12073, 12074, 12075, 12076, 12077, 12078, 12079, 12080, 12081, 12082, 12083, 12084, 12085, 12086, 12087, 12088, 12089, 12090, 12091, 12092, 12093, 12094, 12095, 12096, 12097, 12098, 12099, 12100, 12101, 12102, 12103, 12104, 12105, 12106, 12107, 12108, 12109, 12110, 12111, 12112, 12113