Feature Engineering 101

Topic - 7

Handling Mixed & Date-Time Variables

Handling Mixed Data in Machine Learning

In [1]:
```python
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('titanic.csv')
```

In [3]:
```
df.head()
```

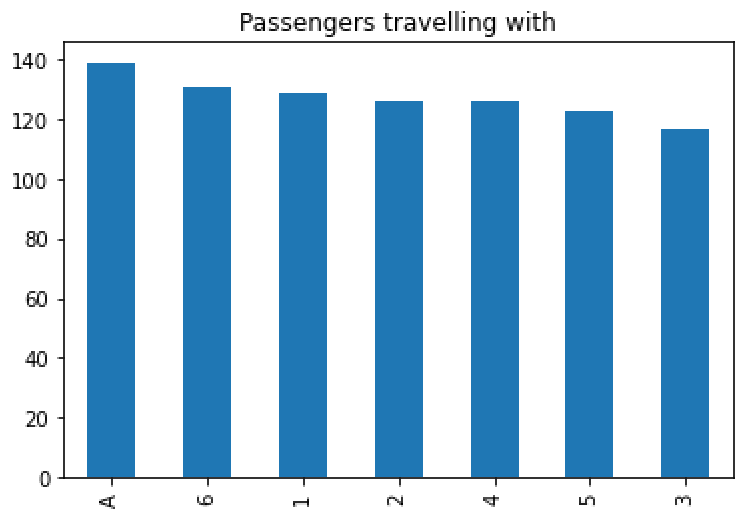
Out[3]:

	Cabin	Ticket	number	Survived
0	NaN	A/5 21171	5	0
1	C85	PC 17599	3	1
2	NaN	STON/O2. 3101282	6	1
3	C123	113803	3	1
4	NaN	373450	A	0

In [4]:
```
df['number'].unique()
```

Out[4]:
```
array(['5', '3', '6', 'A', '2', '1', '4'], dtype=object)
```

In [5]:
```
import matplotlib.pylab as plt
fig = df['number'].value_counts().plot.bar()
fig.set_title('Passengers travelling with')
plt.show()
```



Extract numerical part and Catagorical part

In [6]:
```
df['number_numerical'] = pd.to_numeric(df["number"],errors='coerce',downcast='integer')

df['number_categorical'] = np.where(df['number_numerical'].isnull(),df['number'],np.nan)

df.head()
```

Out[6]:

	Cabin	Ticket	number	Survived	number_numerical	number_categorical
0	NaN	A/5 21171	5	0	5.0	NaN
1	C85	PC 17599	3	1	3.0	NaN
2	NaN	STON/O2. 3101282	6	1	6.0	NaN

	Cabin	Ticket	number	Survived	number_numerical	number_categorical
3	C123	113803	3	1	3.0	NaN
4	NaN	373450	A	0	NaN	A

In [7]:

```
df['Cabin'].unique()
```

Out[7]:

```
array([nan, 'C85', 'C123', 'E46', 'G6', 'C103', 'D56', 'A6',
       'C23 C25 C27', 'B78', 'D33', 'B30', 'C52', 'B28', 'C83', 'F33',
       'F G73', 'E31', 'A5', 'D10 D12', 'D26', 'C110', 'B58 B60', 'E101',
       'F E69', 'D47', 'B86', 'F2', 'C2', 'E33', 'B19', 'A7', 'C49', 'F4',
       'A32', 'B4', 'B80', 'A31', 'D36', 'D15', 'C93', 'C78', 'D35',
       'C87', 'B77', 'E67', 'B94', 'C125', 'C99', 'C118', 'D7', 'A19',
       'B49', 'D', 'C22 C26', 'C106', 'C65', 'E36', 'C54',
       'B57 B59 B63 B66', 'C7', 'E34', 'C32', 'B18', 'C124', 'C91', 'E40',
       'T', 'C128', 'D37', 'B35', 'E50', 'C82', 'B96 B98', 'E10', 'E44',
       'A34', 'C104', 'C111', 'C92', 'E38', 'D21', 'E12', 'E63', 'A14',
       'B37', 'C30', 'D20', 'B79', 'E25', 'D46', 'B73', 'C95', 'B38',
       'B39', 'B22', 'C86', 'C70', 'A16', 'C101', 'C68', 'A10', 'E68',
       'B41', 'A20', 'D19', 'D50', 'D9', 'A23', 'B50', 'A26', 'D48',
       'E58', 'C126', 'B71', 'B51 B53 B55', 'D49', 'B5', 'B20', 'F G63',
       'C62 C64', 'E24', 'C90', 'C45', 'E8', 'B101', 'D45', 'C46', 'D30',
       'E121', 'D11', 'E77', 'F38', 'B3', 'D6', 'B82 B84', 'D17', 'A36',
       'B102', 'B69', 'E49', 'C47', 'D28', 'E17', 'A24', 'C50', 'B42',
       'C148'], dtype=object)
```

In [8]:

```
df['Ticket'].unique()
```

Out[8]:

```
array(['A/5 21171', 'PC 17599', 'STON/O2. 3101282', '113803', '373450',
       '330877', '17463', '349909', '347742', '237736', 'PP 9549',
       '113783', 'A/5. 2151', '347082', '350406', '248706', '382652',
       '244373', '345763', '2649', '239865', '248698', '330923', '113788',
       '347077', '2631', '19950', '330959', '349216', 'PC 17601',
       'PC 17569', '335677', 'C.A. 24579', 'PC 17604', '113789', '2677',
       'A./5. 2152', '345764', '2651', '7546', '11668', '349253',
       'SC/Paris 2123', '330958', 'S.C./A.4. 23567', '370371', '14311',
       '2662', '349237', '3101295', 'A/4. 39886', 'PC 17572', '2926',
       '113509', '19947', 'C.A. 31026', '2697', 'C.A. 34651', 'CA 2144',
       '2669', '113572', '36973', '347088', 'PC 17605', '2661',
       'C.A. 29395', 'S.P. 3464', '3101281', '315151', 'C.A. 33111',
       'S.O.C. 14879', '2680', '1601', '348123', '349208', '374746',
       '248738', '364516', '345767', '345779', '330932', '113059',
       'SO/C 14885', '3101278', 'W./C. 6608', 'SOTON/OQ 392086', '343275',
       '343276', '347466', 'W.E.P. 5734', 'C.A. 2315', '364500', '374910',
       'PC 17754', 'PC 17759', '231919', '244367', '349245', '349215',
       '35281', '7540', '3101276', '349207', '343120', '312991', '349249',
       '371110', '110465', '2665', '324669', '4136', '2627',
       'STON/O 2. 3101294', '370369', 'PC 17558', 'A4. 54510', '27267',
       '370372', 'C 17369', '2668', '347061', '349241',
       'SOTON/O.Q. 3101307', 'A/5. 3337', '228414', 'C.A. 29178',
       'SC/PARIS 2133', '11752', '7534', 'PC 17593', '2678', '347081',
       'STON/O2. 3101279', '365222', '231945', 'C.A. 33112', '350043',
       '230080', '244310', 'S.O.P. 1166', '113776', 'A.5. 11206',
       'A/5. 851', 'Fa 265302', 'PC 17597', '35851', 'SOTON/OQ 392090',
       '315037', 'CA. 2343', '371362', 'C.A. 33595', '347068', '315093',
       '363291', '113505', 'PC 17318', '111240', 'STON/O 2. 3101280',
       '17764', '350404', '4133', 'PC 17595', '250653', 'LINE',
       'SC/PARIS 2131', '230136', '315153', '113767', '370365', '111428',
       '364849', '349247', '234604', '28424', '350046', 'PC 17610',
       '368703', '4579', '370370', '248747', '345770', '3101264', '2628',
       'A/5 3540', '347054', '2699', '367231', '112277',
       'SOTON/O.Q. 3101311', 'F.C.C. 13528', 'A/5 21174', '250646',
'367229', '35273', 'STON/O2. 3101283', '243847', '11813',
'W/C 14208', 'SOTON/OQ 392089', '220367', '21440', '349234',
'19943', 'PP 4348', 'SW/PP 751', 'A/5 21173', '236171', '347067',
'237442', 'C.A. 29566', 'W./C. 6609', '26707', 'C.A. 31921',
'28665', 'SCO/W 1585', '367230', 'W./C. 14263',
'STON/O 2. 3101275', '2694', '19928', '347071', '250649', '11751',
'244252', '362316', '113514', 'A/5. 3336', '370129', '2650',
'PC 17585', '110152', 'PC 17755', '230433', '384461', '110413',
'112059', '382649', 'C.A. 17248', '347083', 'PC 17582', 'PC 17760',
'113798', '250644', 'PC 17596', '370375', '13502', '347073',
'239853', 'C.A. 2673', '336439', '347464', '345778', 'A/5. 10482',
'113056', '349239', '345774', '349206', '237798', '370373',
'19877', '11967', 'SC/Paris 2163', '349236', '349233', 'PC 17612',
'2693', '113781', '19988', '9234', '367226', '226593', 'A/5 2466',
'17421', 'PC 17758', 'P/PP 3381', 'PC 17485', '11767', 'PC 17608',
'250651', '349243', 'F.C.C. 13529', '347470', '29011', '36928',
'16966', 'A/5 21172', '349219', '234818', '345364', '28551',
'111361', '113043', 'PC 17611', '349225', '7598', '113784',
'248740', '244361', '229236', '248733', '31418', '386525',
'C.A. 37671', '315088', '7267', '113510', '2695', '2647', '345783',
'237671', '330931', '330980', 'SC/PARIS 2167', '2691',
'SOTON/O.Q. 3101310', 'C 7076', '110813', '2626', '14313',
'PC 17477', '11765', '3101267', '323951', 'C 7077', '113503',
'2648', '347069', 'PC 17757', '2653', 'STON/O 2. 3101293',
'349227', '27849', '367655', 'SC 1748', '113760', '350034',
'3101277', '350052', '350407', '28403', '244278', '240929',
'STON/O 2. 3101289', '341826', '4137', '315096', '28664', '347064',
'29106', '312992', '349222', '394140', 'STON/O 2. 3101269',
'343095', '28220', '250652', '28228', '345773', '349254',
'A/5. 13032', '315082', '347080', 'A/4. 34244', '2003', '250655',
'364851', 'SOTON/O.Q. 392078', '110564', '376564', 'SC/AH 3085',
'STON/O 2. 3101274', '13507', 'C.A. 18723', '345769', '347076',
'230434', '65306', '33638', '113794', '2666', '113786', '65303',
'113051', '17453', 'A/5 2817', '349240', '13509', '17464',
'F.C.C. 13531', '371060', '19952', '364506', '111320', '234360',
'A/S 2816', 'SOTON/O.Q. 3101306', '113792', '36209', '323592',
'315089', 'SC/AH Basle 541', '7553', '31027', '3460', '350060',
'3101298', '239854', 'A/5 3594', '4134', '11771', 'A.5. 18509',
'65304', 'SOTON/OQ 3101317', '113787', 'PC 17609', 'A/4 45380',
'36947', 'C.A. 6212', '350035', '315086', '364846', '330909',
'4135', '26360', '111427', 'C 4001', '382651', 'SOTON/OQ 3101316',
'PC 17473', 'PC 17603', '349209', '36967', 'C.A. 34260', '226875',
'349242', '12749', '349252', '2624', '2700', '367232',
'W./C. 14258', 'PC 17483', '3101296', '29104', '2641', '2690',
'315084', '113050', 'PC 17761', '364498', '13568', 'WE/P 5735',
'2908', '693', 'SC/PARIS 2146', '244358', '330979', '2620',
'347085', '113807', '11755', '345572', '372622', '349251',
'218629', 'SOTON/OQ 392082', 'SOTON/O.Q. 392087', 'A/4 48871',
'349205', '2686', '350417', 'S.W./PP 752', '11769', 'PC 17474',
'14312', 'A/4. 20589', '358585', '243880', '2689',
'STON/O 2. 3101286', '237789', '13049', '3411', '237565', '13567',
'14973', 'A./5. 3235', 'STON/O 2. 3101273', 'A/5 3902', '364848',
'SC/AH 29037', '248727', '2664', '349214', '113796', '364511',
'111426', '349910', '349246', '113804', 'SOTON/O.Q. 3101305',
'370377', '364512', '220845', '31028', '2659', '11753', '350029',
'54636', '36963', '219533', '349224', '334912', '27042', '347743',
'13214', '112052', '237668', 'STON/O 2. 3101292', '350050',
'349231', '13213', 'S.O./P.P. 751', 'CA. 2314', '349221', '8475',
'330919', '365226', '349223', '29751', '2623', '5727', '349210',
'STON/O 2. 3101285', '234686', '312993', 'A/5 3536', '19996',
'29750', 'F.C. 12750', 'C.A. 24580', '244270', '239856', '349912',
'342826', '4138', '330935', '6563', '349228', '350036', '24160',
'17474', '349256', '2672', '113800', '248731', '363592', '35852',
'348121', 'PC 17475', '36864', '350025', '223596', 'PC 17476',
'PC 17482', '113028', '7545', '250647', '348124', '34218', '36568',
'347062', '350048', '12233', '250643', '113806', '315094', '36866',
'236853', 'STON/O2. 3101271', '239855', '28425', '233639',
'349201', '349218', '16988', '376566', 'STON/O 2. 3101288',
'250648', '113773', '335097', '29103', '392096', '345780',
'349204', '350042', '29108', '363294', 'SOTON/O2 3101272', '2663',
'347074', '112379', '364850', '8471', '345781', '350047',
'S.O./P.P. 3', '2674', '29105', '347078', '383121', '36865',
'2687', '113501', 'W./C. 6607', 'SOTON/O.Q. 3101312', '374887',
'3101265', '12460', 'PC 17600', '349203', '28213', '17465',
'349244', '2685', '2625', '347089', '347063', '112050', '347087',
'248723', '3474', '28206', '364499', '112058', 'STON/O2. 3101290',
'S.C./PARIS 2079', 'C 7075', '315098', '19972', '368323', '367228',
'2671', '347468', '2223', 'PC 17756', '315097', '392092', '11774',
'SOTON/O2 3101287', '2683', '315090', 'C.A. 5547', '349213',
'347060', 'PC 17592', '392091', '113055', '2629', '350026',
'28134', '17466', '233866', '236852', 'SC/PARIS 2149', 'PC 17590',
'345777', '349248', '695', '345765', '2667', '349212', '349217',
'349257', '7552', 'C.A./SOTON 34068', 'SOTON/OQ 392076', '211536',
'112053', '111369', '370376'], dtype=object)
```

In [9]:

```
df['cabin_num'] = df['Cabin'].str.extract('(\d+)') # captures numerical part
df['cabin_cat'] = df['Cabin'].str[0] # captures the first letter

df.head()
```

Out[9]:

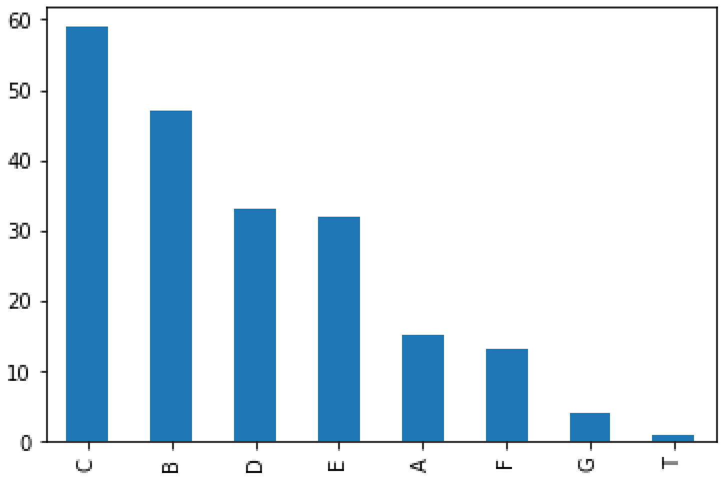
	Cabin	Ticket	number	Survived	number_numerical	number_categorical	cabin_num	cabin_cat
0	NaN	A/5 21171	5	0	5.0	NaN	NaN	NaN
1	C85	PC 17599	3	1	3.0	NaN	85	C
2	NaN	STON/O2. 3101282	6	1	6.0	NaN	NaN	NaN
3	C123	113803	3	1	3.0	NaN	123	C
4	NaN	373450	A	0	NaN	A	NaN	NaN

In [10]:

```
df['cabin_cat'].value_counts().plot(kind='bar')
```

Out[10]:

```
<AxesSubplot:>
```



In [11]:

```
# extract the last bit of ticket as number
df['ticket_num'] = df['Ticket'].apply(lambda s: s.split()[-1])
df['ticket_num'] = pd.to_numeric(df['ticket_num'],
                                 errors='coerce',
                                 downcast='integer')

# extract the first part of ticket as category
```

```
df['ticket_cat'] = df['Ticket'].apply(lambda s: s.split()[0])
df['ticket_cat'] = np.where(df['ticket_cat'].str.isdigit(), np.nan,
                            df['ticket_cat'])

df.head(20)
```

Out[11]:

	Cabin	Ticket	number	Survived	number_numerical	number_categorical	cabin_num	cabin_cat	ticket_num
0	NaN	A/5 21171	5	0	5.0	NaN	NaN	NaN	21171.0
1	C85	PC 17599	3	1	3.0	NaN	85	C	17599.0
2	NaN	STON/O2. 3101282	6	1	6.0	NaN	NaN	NaN	3101282.0
3	C123	113803	3	1	3.0	NaN	123	C	113803.0
4	NaN	373450	A	0	NaN	A	NaN	NaN	373450.0
5	NaN	330877	2	0	2.0	NaN	NaN	NaN	330877.0
6	E46	17463	2	0	2.0	NaN	46	E	17463.0
7	NaN	349909	5	0	5.0	NaN	NaN	NaN	349909.0
8	NaN	347742	1	1	1.0	NaN	NaN	NaN	347742.0
9	NaN	237736	A	1	NaN	A	NaN	NaN	237736.0
10	G6	PP 9549	1	1	1.0	NaN	6	G	9549.0
11	C103	113783	1	1	1.0	NaN	103	C	113783.0
12	NaN	A/5. 2151	3	0	3.0	NaN	NaN	NaN	2151.0
13	NaN	347082	3	0	3.0	NaN	NaN	NaN	347082.0
14	NaN	350406	5	0	5.0	NaN	NaN	NaN	350406.0
15	NaN	248706	3	1	3.0	NaN	NaN	NaN	248706.0
16	NaN	382652	3	0	3.0	NaN	NaN	NaN	382652.0
17	NaN	244373	2	1	2.0	NaN	NaN	NaN	244373.0
18	NaN	345763	5	0	5.0	NaN	NaN	NaN	345763.0
19	NaN	2649	4	1	4.0	NaN	NaN	NaN	2649.0

In [12]:

```
df['ticket_cat'].unique()
```

Out[12]:

```
array(['A/5', 'PC', 'STON/O2.', nan, 'PP', 'A/5.', 'C.A.', 'A./5.',
       'SC/Paris', 'S.C./A.4.', 'A/4.', 'CA', 'S.P.', 'S.O.C.', 'SO/C',
       'W./C.', 'SOTON/OQ', 'W.E.P.', 'STON/O', 'A4.', 'C', 'SOTON/O.Q.',
       'SC/PARIS', 'S.O.P.', 'A.5.', 'Fa', 'CA.', 'LINE', 'F.C.C.', 'W/C',
       'SW/PP', 'SCO/W', 'P/PP', 'SC', 'SC/AH', 'A/S', 'A/4', 'WE/P',
       'S.W./PP', 'S.O./P.P.', 'F.C.', 'SOTON/O2', 'S.C./PARIS',
       'C.A./SOTON'], dtype=object)
```

In [13]:

```
df['ticket_num'].unique()
```

Out[13]:

```
array([2.117100e+04, 1.759900e+04, 3.101282e+06, 1.138030e+05,
       3.734500e+05, 3.308770e+05, 1.746300e+04, 3.499090e+05,
       3.477420e+05, 2.377360e+05, 9.549000e+03, 1.137830e+05,
       2.151000e+03, 3.470820e+05, 3.504060e+05, 2.487060e+05,
       3.826520e+05, 2.443730e+05, 3.457630e+05, 2.649000e+03,
       2.398650e+05, 2.486980e+05, 3.309230e+05, 1.137880e+05,
       3.470770e+05, 2.631000e+03, 1.995000e+04, 3.309590e+05,
       3.492160e+05, 1.760100e+04, 1.756900e+04, 3.356770e+05,
       2.457900e+04, 1.760400e+04, 1.137890e+05, 2.677000e+03,
       2.152000e+03, 3.457640e+05, 2.651000e+03, 7.546000e+03,
       1.166800e+04, 3.492530e+05, 2.123000e+03, 3.309580e+05,
       2.356700e+04, 3.703710e+05, 1.431100e+04, 2.662000e+03,
       3.492370e+05, 3.101295e+06, 3.988600e+04, 1.757200e+04,
       2.926000e+03, 1.135090e+05, 1.994700e+04, 3.102600e+04,
       2.697000e+03, 3.465100e+04, 2.144000e+03, 2.669000e+03,
       1.135720e+05, 3.697300e+04, 3.470880e+05, 1.760500e+04,
       2.661000e+03, 2.939500e+04, 3.464000e+03, 3.101281e+06,
       3.151510e+05, 3.311100e+04, 1.487900e+04, 2.680000e+03,
       1.601000e+03, 3.481230e+05, 3.492080e+05, 3.747460e+05,
       2.487380e+05, 3.645160e+05, 3.457670e+05, 3.457790e+05,
       3.309320e+05, 1.130590e+05, 1.488500e+04, 3.101278e+06,
       6.608000e+03, 3.920860e+05, 3.432750e+05, 3.432760e+05,
       3.474660e+05, 5.734000e+03, 2.315000e+03, 3.645000e+05,
       3.749100e+05, 1.775400e+04, 1.775900e+04, 2.319190e+05,
       2.443670e+05, 3.492450e+05, 3.492150e+05, 3.528100e+04,
       7.540000e+03, 3.101276e+06, 3.492070e+05, 3.431200e+05,
       3.129910e+05, 3.492490e+05, 3.711100e+05, 1.104650e+05,
       2.665000e+03, 3.246690e+05, 4.136000e+03, 2.627000e+03,
       3.101294e+06, 3.703690e+05, 1.755800e+04, 5.451000e+04,
       2.726700e+04, 3.703720e+05, 1.736900e+04, 2.668000e+03,
       3.470610e+05, 3.492410e+05, 3.101307e+06, 3.337000e+03,
       2.284140e+05, 2.917800e+04, 2.133000e+03, 1.175200e+04,
       7.534000e+03, 1.759300e+04, 2.678000e+03, 3.470810e+05,
       3.101279e+06, 3.652220e+05, 2.319450e+05, 3.311200e+04,
       3.500430e+05, 2.300800e+05, 2.443100e+05, 1.166000e+03,
       1.137760e+05, 1.120600e+04, 8.510000e+02, 2.653020e+05,
       1.759700e+04, 3.585100e+04, 3.920900e+05, 3.150370e+05,
       2.343000e+03, 3.713620e+05, 3.359500e+04, 3.470680e+05,
       3.150930e+05, 3.632910e+05, 1.135050e+05, 1.731800e+04,
       1.112400e+05, 3.101280e+06, 1.776400e+04, 3.504040e+05,
       4.133000e+03, 1.759500e+04, 2.506530e+05,          nan,
       2.131000e+03, 2.301360e+05, 3.151530e+05, 1.137670e+05,
       3.703650e+05, 1.114280e+05, 3.648490e+05, 3.492470e+05,
       2.346040e+05, 2.842400e+04, 3.500460e+05, 1.761000e+04,
       3.687030e+05, 4.579000e+03, 3.703700e+05, 2.487470e+05,
       3.457700e+05, 3.101264e+06, 2.628000e+03, 3.540000e+03,
       3.470540e+05, 2.699000e+03, 3.672310e+05, 1.122770e+05,
       3.101311e+06, 1.352800e+04, 2.117400e+04, 2.506460e+05,
       3.672290e+05, 3.527300e+04, 3.101283e+06, 2.438470e+05,
       1.181300e+04, 1.420800e+04, 3.920890e+05, 2.203670e+05,
       2.144000e+04, 3.492340e+05, 1.994300e+04, 4.348000e+03,
       7.510000e+02, 2.117300e+04, 2.361710e+05, 3.470670e+05,
       2.374420e+05, 2.956600e+04, 6.609000e+03, 2.670700e+04,
       3.192100e+04, 2.866500e+04, 1.585000e+03, 3.672300e+05,
       1.426300e+04, 3.101275e+06, 2.694000e+03, 1.992800e+04,
       3.470710e+05, 2.506490e+05, 1.175100e+04, 2.442520e+05,
       3.623160e+05, 1.135140e+05, 3.336000e+03, 3.701290e+05,
       2.650000e+03, 1.758500e+04, 1.101520e+05, 1.775500e+04,
       2.304330e+05, 3.844610e+05, 1.104130e+05, 1.120590e+05,
       3.826490e+05, 1.724800e+04, 3.470830e+05, 1.758200e+04,
       1.776000e+04, 1.137980e+05, 2.506440e+05, 1.759600e+04,
       3.703750e+05, 1.350200e+04, 3.470730e+05, 2.398530e+05,
       2.673000e+03, 3.364390e+05, 3.474640e+05, 3.457780e+05,
       1.048200e+04, 1.130560e+05, 3.492390e+05, 3.457740e+05,
       3.492060e+05, 2.377980e+05, 3.703730e+05, 1.987700e+04,
       1.196700e+04, 2.163000e+03, 3.492360e+05, 3.492330e+05,
       1.761200e+04, 2.693000e+03, 1.137810e+05, 1.998800e+04,
       9.234000e+03, 3.672260e+05, 2.265930e+05, 2.466000e+03,
       1.742100e+04, 1.775800e+04, 3.381000e+03, 1.748500e+04,
       1.176700e+04, 1.760800e+04, 2.506510e+05, 3.492430e+05,
       1.352900e+04, 3.474700e+05, 2.901100e+04, 3.692800e+04,
       1.696600e+04, 2.117200e+04, 3.492190e+05, 2.348180e+05,
       3.453640e+05, 2.855100e+04, 1.113610e+05, 1.130430e+05,
       1.761100e+04, 3.492250e+05, 7.598000e+03, 1.137840e+05,
       2.487400e+05, 2.443610e+05, 2.292360e+05, 2.487330e+05,
       3.141800e+04, 3.865250e+05, 3.767100e+04, 3.150880e+05,
       7.267000e+03, 1.135100e+05, 2.695000e+03, 2.647000e+03,
       3.457830e+05, 2.376710e+05, 3.309310e+05, 3.309800e+05,
       2.167000e+03, 2.691000e+03, 3.101310e+06, 7.076000e+03,
       1.108130e+05, 2.626000e+03, 1.431300e+04, 1.747700e+04,
       1.176500e+04, 3.101267e+06, 3.239510e+05, 7.077000e+03,
       1.135030e+05, 2.648000e+03, 3.470690e+05, 1.775700e+04,
       2.653000e+03, 3.101293e+06, 3.492270e+05, 2.784900e+04,
       3.676550e+05, 1.748000e+03, 1.137600e+05, 3.500340e+05,
       3.101277e+06, 3.500520e+05, 3.504070e+05, 2.840300e+04,
       2.442780e+05, 2.409290e+05, 3.101289e+06, 3.418260e+05,
       4.137000e+03, 3.150960e+05, 2.866400e+04, 3.470640e+05,
       2.910600e+04, 3.129920e+05, 3.492220e+05, 3.941400e+05,
       3.101269e+06, 3.430950e+05, 2.822000e+04, 2.506520e+05,
       2.822800e+04, 3.457730e+05, 3.492540e+05, 1.303200e+04,
       3.150820e+05, 3.470800e+05, 3.424400e+04, 2.003000e+03,
       2.506550e+05, 3.648510e+05, 3.920780e+05, 1.105640e+05,
       3.765640e+05, 3.085000e+03, 3.101274e+06, 1.350700e+04,
       1.872300e+04, 3.457690e+05, 3.470760e+05, 2.304340e+05,
       6.530600e+04, 3.363800e+04, 1.137940e+05, 2.666000e+03,
       1.137860e+05, 6.530300e+04, 1.130510e+05, 1.745300e+04,
       2.817000e+03, 3.492400e+05, 1.350900e+04, 1.746400e+04,
       1.353100e+04, 3.710600e+05, 1.995200e+04, 3.645060e+05,
       1.113200e+05, 2.343600e+05, 2.816000e+03, 3.101306e+06,
       1.137920e+05, 3.620900e+04, 3.235920e+05, 3.150890e+05,
       5.410000e+02, 7.553000e+03, 3.102700e+04, 3.460000e+03,
       3.500600e+05, 3.101298e+06, 2.398540e+05, 3.594000e+03,
       4.134000e+03, 1.177100e+04, 1.850900e+04, 6.530400e+04,
       3.101317e+06, 1.137870e+05, 1.760900e+04, 4.538000e+04,
       3.694700e+04, 6.212000e+03, 3.500350e+05, 3.150860e+05,
       3.648460e+05, 3.309090e+05, 4.135000e+03, 2.636000e+04,
       1.114270e+05, 4.001000e+03, 3.826510e+05, 3.101316e+06,
       1.747300e+04, 1.760300e+04, 3.492090e+05, 3.696700e+04,
       3.426000e+04, 2.268750e+05, 3.492420e+05, 1.274900e+04,
       3.492520e+05, 2.624000e+03, 2.700000e+03, 3.672320e+05,
       1.425800e+04, 1.748300e+04, 3.101296e+06, 2.910400e+04,
       2.641000e+03, 2.690000e+03, 3.150840e+05, 1.130500e+05,
       1.776100e+04, 3.644980e+05, 1.356800e+04, 5.735000e+03,
       2.908000e+03, 6.930000e+02, 2.146000e+03, 2.443580e+05,
       3.309790e+05, 2.620000e+03, 3.470850e+05, 1.138070e+05,
       1.175500e+04, 3.455720e+05, 3.726220e+05, 3.492510e+05,
       2.186290e+05, 3.920820e+05, 3.920870e+05, 4.887100e+04,
       3.492050e+05, 2.686000e+03, 3.504170e+05, 7.520000e+02,
       1.176900e+04, 1.747400e+04, 1.431200e+04, 2.058900e+04,
       3.585850e+05, 2.438800e+05, 2.689000e+03, 3.101286e+06,
       2.377890e+05, 1.304900e+04, 3.411000e+03, 2.375650e+05,
       1.356700e+04, 1.497300e+04, 3.235000e+03, 3.101273e+06,
       3.902000e+03, 3.648480e+05, 2.903700e+04, 2.487270e+05,
       2.664000e+03, 3.492140e+05, 1.137960e+05, 3.645110e+05,
       1.114260e+05, 3.499100e+05, 3.492460e+05, 1.138040e+05,
       3.101305e+06, 3.703770e+05, 3.645120e+05, 2.208450e+05,
       3.102800e+04, 2.659000e+03, 1.175300e+04, 3.500290e+05,
       5.463600e+04, 3.696300e+04, 2.195330e+05, 3.492240e+05,
       3.349120e+05, 2.704200e+04, 3.477430e+05, 1.321400e+04,
       1.120520e+05, 2.376680e+05, 3.101292e+06, 3.500500e+05,
       3.492310e+05, 1.321300e+04, 2.314000e+03, 3.492210e+05,
       8.475000e+03, 3.309190e+05, 3.652260e+05, 3.492230e+05,
       2.975100e+04, 2.623000e+03, 5.727000e+03, 3.492100e+05,
       3.101285e+06, 2.346860e+05, 3.129930e+05, 3.536000e+03,
       1.999600e+04, 2.975000e+04, 1.275000e+04, 2.458000e+04,
       2.442700e+05, 2.398560e+05, 3.499120e+05, 3.428260e+05,
       4.138000e+03, 3.309350e+05, 6.563000e+03, 3.492280e+05,
       3.500360e+05, 2.416000e+04, 3.492560e+05, 2.672000e+03,
       1.138000e+05, 2.487310e+05, 3.635920e+05, 3.585200e+04,
       3.481210e+05, 1.747500e+04, 3.686400e+04, 3.500250e+05,
       2.235960e+05, 1.747600e+04, 1.748200e+04, 1.130280e+05,
       7.545000e+03, 2.506470e+05, 3.481240e+05, 3.421800e+04,
       3.656800e+04, 3.470620e+05, 3.500480e+05, 1.223300e+04,
       2.506430e+05, 1.138060e+05, 3.150940e+05, 3.686600e+04,
       2.368530e+05, 3.101271e+06, 2.398550e+05, 2.842500e+04,
       2.336390e+05, 3.492010e+05, 3.492180e+05, 1.698800e+04,
       3.765660e+05, 3.101288e+06, 2.506480e+05, 1.137730e+05,
       3.350970e+05, 2.910300e+04, 3.920960e+05, 3.457800e+05,
       3.492040e+05, 3.500420e+05, 2.910800e+04, 3.632940e+05,
       3.101272e+06, 2.663000e+03, 3.470740e+05, 1.123790e+05,
       3.648500e+05, 8.471000e+03, 3.457810e+05, 3.500470e+05,
       3.000000e+00, 2.674000e+03, 2.910500e+04, 3.470780e+05,
       3.831210e+05, 3.686500e+04, 2.687000e+03, 1.135010e+05,
       6.607000e+03, 3.101312e+06, 3.748870e+05, 3.101265e+06,
       1.246000e+04, 1.760000e+04, 3.492030e+05, 2.821300e+04,
       1.746500e+04, 3.492440e+05, 2.685000e+03, 2.625000e+03,
       3.470890e+05, 3.470630e+05, 1.120500e+05, 3.470870e+05,
       2.487230e+05, 3.474000e+03, 2.820600e+04, 3.644990e+05,
       1.120580e+05, 3.101290e+06, 2.079000e+03, 7.075000e+03,
       3.150980e+05, 1.997200e+04, 3.683230e+05, 3.672280e+05,
       2.671000e+03, 3.474680e+05, 2.223000e+03, 1.775600e+04,
       3.150970e+05, 3.920920e+05, 1.177400e+04, 3.101287e+06,
       2.683000e+03, 3.150900e+05, 5.547000e+03, 3.492130e+05,
       3.470600e+05, 1.759200e+04, 3.920910e+05, 1.130550e+05,
       2.629000e+03, 3.500260e+05, 2.813400e+04, 1.746600e+04,
       2.338660e+05, 2.368520e+05, 2.149000e+03, 1.759000e+04,
       3.457770e+05, 3.492480e+05, 6.950000e+02, 3.457650e+05,
       2.667000e+03, 3.492120e+05, 3.492170e+05, 3.492570e+05,
       7.552000e+03, 3.406800e+04, 3.920760e+05, 2.115360e+05,
       1.120530e+05, 1.113690e+05, 3.703760e+05])
```

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

Handling Date and Time variable in Machine Learning

In [14]:

```
import numpy as np
import pandas as pd
```

In [15]:

```
date = pd.read_csv('orders.csv')
time = pd.read_csv('messages.csv')
```

In [16]:

```
date.head()
```

Out[16]:

	date	product_id	city_id	orders
0	2019-12-10	5628	25	3
1	2018-08-15	3646	14	157
2	2018-10-23	1859	25	1
3	2019-08-17	7292	25	1

	date	product_id	city_id	orders
4	2019-01-06	4344	25	3

In [17]:
```
time.head()
```

Out[17]:

	date	msg
0	2013-12-15 00:50:00	ищу на сегодня мужика 37
1	2014-04-29 23:40:00	ПАРЕНЬ БИ ИЩЕТ ДРУГА СЕЙЧАС!! СМС ММС 0955532826
2	2012-12-30 00:21:00	Днепр.м 43 позн.с д/ж *.о 067.16.34.576
3	2014-11-28 00:31:00	КИЕВ ИЩУ Д/Ж ДО 45 МНЕ СЕЙЧАС СКУЧНО 093 629 9...
4	2013-10-26 23:11:00	Зая я тебя никогда не обижу люблю тебя!) Даше

In [18]:
```
print(date.info())
print(time.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   date        1000 non-null   object
 1   product_id  1000 non-null   int64 
 2   city_id     1000 non-null   int64 
 3   orders      1000 non-null   int64 
dtypes: int64(3), object(1)
memory usage: 31.4+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   date    1000 non-null   object
 1   msg     1000 non-null   object
dtypes: object(2)
memory usage: 15.8+ KB
None
```

Working with dates

In [19]:
```
# Converting to datetime datatype
date['date'] = pd.to_datetime(date['date'])
```

In [20]:
```
date.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype         
---  ------      --------------  -----         
 0   date        1000 non-null   datetime64[ns]
 1   product_id  1000 non-null   int64         
 2   city_id     1000 non-null   int64         
 3   orders      1000 non-null   int64
dtypes: datetime64[ns](1), int64(3)
memory usage: 31.4 KB
```

In [21]:

```
#Extract year----
date['date_year'] = date['date'].dt.year

#Extract month----
date['date_month_no'] = date['date'].dt.month

#Extract month name----
date['date_month_name'] = date['date'].dt.month_name()

#Extract day----
date['date_day'] = date['date'].dt.day

#Extract day of week----
date['date_dow'] = date['date'].dt.dayofweek

#Extract day name----
date['date_dow_name'] = date['date'].dt.day_name()

#Extract date is weekend?----
date['date_is_weekend'] = np.where(date['date_dow_name'].isin(['Sunday', 'Saturday']), 1,

#Extract date week----
date['date_week'] = date['date'].dt.week

#Extract quarter----
date['quarter'] = date['date'].dt.quarter

#Extract semester----
date['semester'] = np.where(date['quarter'].isin([1,2]), 1, 2)
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_75188/3985301858.py:23: FutureWarning: Series.dt.
weekofyear and Series.dt.week have been deprecated.  Please use Series.dt.isocalendar().we
ek instead.
  date['date_week'] = date['date'].dt.week
```

In [22]:

```
date.drop(columns=['product_id','city_id','orders']).head()
```

Out[22]:

	date	date_year	date_month_no	date_month_name	date_day	date_dow	date_dow_name	date_is_weekend	dat
0	2019-12-10	2019	12	December	10	1	Tuesday	0	
1	2018-08-15	2018	8	August	15	2	Wednesday	0	
2	2018-10-23	2018	10	October	23	1	Tuesday	0	
3	2019-08-17	2019	8	August	17	5	Saturday	1	
4	2019-01-06	2019	1	January	6	6	Sunday	1	

Working with Times

In [23]:

```
import datetime
```

```
today = datetime.datetime.today()

today
```

Out[23]:
```
datetime.datetime(2023, 1, 29, 23, 41, 23, 626953)
```

In [24]:
```
today - date['date']
```

Out[24]:
```
0      1146 days 23:41:23.626953
1      1628 days 23:41:23.626953
2      1559 days 23:41:23.626953
3      1261 days 23:41:23.626953
4      1484 days 23:41:23.626953
                  ...
995    1574 days 23:41:23.626953
996    1515 days 23:41:23.626953
997    1363 days 23:41:23.626953
998    1428 days 23:41:23.626953
999    1202 days 23:41:23.626953
Name: date, Length: 1000, dtype: timedelta64[ns]
```

In [25]:
```
(today - date['date']).dt.days
```

Out[25]:
```
0       1146
1       1628
2       1559
3       1261
4       1484
        ...
995     1574
996     1515
997     1363
998     1428
999     1202
Name: date, Length: 1000, dtype: int64
```

In [26]:
```
# Months passed

np.round((today -date['date']) / np.timedelta64(1, 'M'),0)
```

Out[26]:
```
0       38.0
1       54.0
2       51.0
3       41.0
4       49.0
        ...
995     52.0
996     50.0
997     45.0
998     47.0
999     40.0
Name: date, Length: 1000, dtype: float64
```

In [27]:
```
time.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   date    1000 non-null   object
 1   msg     1000 non-null   object
dtypes: object(2)
memory usage: 15.8+ KB
```

In [28]:

```
# Converting to datetime datatype
time['date'] = pd.to_datetime(time['date'])
```

In [29]:

```
time['hour'] = time['date'].dt.hour
time['min'] = time['date'].dt.minute
time['sec'] = time['date'].dt.second

time.head()
```

Out[29]:

	date	msg	hour	min	sec
0	2013-12-15 00:50:00	ищу на сегодня мужика 37	0	50	0
1	2014-04-29 23:40:00	ПАРЕНЬ БИ ИЩЕТ ДРУГА СЕЙЧАС!! СМС ММС 0955532826	23	40	0
2	2012-12-30 00:21:00	Днепр.м 43 позн.с д/ж *.о 067.16.34.576	0	21	0
3	2014-11-28 00:31:00	КИЕВ ИЩУ Д/Ж ДО 45 МНЕ СЕЙЧАС СКУЧНО 093 629 9...	0	31	0
4	2013-10-26 23:11:00	Зая я тебя никогда не обижу люблю тебя!) Даше	23	11	0

In [30]:

```
#Extract time part
time['time'] = time['date'].dt.time

time.head()
```

Out[30]:

	date	msg	hour	min	sec	time
0	2013-12-15 00:50:00	ищу на сегодня мужика 37	0	50	0	00:50:00
1	2014-04-29 23:40:00	ПАРЕНЬ БИ ИЩЕТ ДРУГА СЕЙЧАС!! СМС ММС 0955532826	23	40	0	23:40:00
2	2012-12-30 00:21:00	Днепр.м 43 позн.с д/ж *.о 067.16.34.576	0	21	0	00:21:00
3	2014-11-28 00:31:00	КИЕВ ИЩУ Д/Ж ДО 45 МНЕ СЕЙЧАС СКУЧНО 093 629 9...	0	31	0	00:31:00
4	2013-10-26 23:11:00	Зая я тебя никогда не обижу люблю тебя!) Даше	23	11	0	23:11:00

In [31]:

```
#Time diff.
today - time['date']
```

Out[31]:

```
0      3332 days 22:51:23.626953
1      3197 days 00:01:23.626953
2      3682 days 23:20:23.626953
3      2984 days 23:10:23.626953
4      3382 days 00:30:23.626953
                  ...
995    3971 days 22:51:23.626953
996    3293 days 00:27:23.626953
997    3758 days 00:04:23.626953
998    3874 days 00:07:23.626953
999    3146 days 00:16:23.626953
Name: date, Length: 1000, dtype: timedelta64[ns]
```

In [32]:

```
# in seconds

(today - time['date'])/np.timedelta64(1,'s')
```

```
Out[32]:
0      2.879671e+08
1      2.762209e+08
2      3.182088e+08
3      2.579010e+08
4      2.922066e+08
           ...
995    3.431767e+08
996    2.845168e+08
997    3.246915e+08
998    3.347140e+08
999    2.718154e+08
Name: date, Length: 1000, dtype: float64
```

```
In [33]:
# in hours

(today - time['date'])/np.timedelta64(1,'h')
```

```
Out[33]:
0      79990.856563
1      76728.023230
2      88391.339896
3      71639.173230
4      81168.506563
           ...
995    95326.856563
996    79032.456563
997    90192.073230
998    92976.123230
999    75504.273230
Name: date, Length: 1000, dtype: float64
```

```
In [ ]:
```