```
## INTRODUCTION TO PANDAS

import pandas as pd
import numpy as np
```

```
!gdown 1E3bwvYGf1ig32RmcYiWc0IXPN-mD_bI_
```

```
Downloading...
From: https://drive.google.com/uc?id=1E3bwvYGf1ig32RmcYiWc0IXPN-mD_bI_
To: /content/mckinsey.csv
100% 83.8k/83.8k [00:00<00:00, 119MB/s]
```

```
df = pd.read_csv("mckinsey.csv")
df
```

	country	year	population	continent	life_exp	gdp_cap
0	Afghanistan	1952	8425333	Asia	28.801	779.445314
1	Afghanistan	1957	9240934	Asia	30.332	820.853030
2	Afghanistan	1962	10267083	Asia	31.997	853.100710
3	Afghanistan	1967	11537966	Asia	34.020	836.197138
4	Afghanistan	1972	13079460	Asia	36.088	739.981106
...	...	...	...	...	...	...
1699	Zimbabwe	1987	9216418	Africa	62.351	706.157306
1700	Zimbabwe	1992	10704340	Africa	60.377	693.420786
1701	Zimbabwe	1997	11404948	Africa	46.809	792.449960
1702	Zimbabwe	2002	11926563	Africa	39.989	672.038623
1703	Zimbabwe	2007	12311143	Africa	43.487	469.709298

1704 rows × 6 columns

```
type(df)
```

```
pandas.core.frame.DataFrame
```

```
type(df["country"])
```

```
pandas.core.series.Series
```

```
type(df[["country"]])
```

```
pandas.core.frame.DataFrame
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1704 entries, 0 to 1703
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   country     1704 non-null   object 
 1   year        1704 non-null   int64  
 2   population  1704 non-null   int64  
 3   continent   1704 non-null   object 
 4   life_exp    1704 non-null   float64
 5   gdp_cap     1704 non-null   float64
dtypes: float64(2), int64(2), object(2)
memory usage: 80.0+ KB
```

```
df.head()
```

	country	year	population	continent	life_exp	gdp_cap
0	Afghanistan	1952	8425333	Asia	28.801	779.445314
1	Afghanistan	1957	9240934	Asia	30.332	820.853030
2	Afghanistan	1962	10267083	Asia	31.997	853.100710
3	Afghanistan	1967	11537966	Asia	34.020	836.197138
4	Afghanistan	1972	13079460	Asia	36.088	739.981106

```
df.head(-10)
```

	country	year	population	continent	life_exp	gdp_cap
0	Afghanistan	1952	8425333	Asia	28.801	779.445314
1	Afghanistan	1957	9240934	Asia	30.332	820.853030
2	Afghanistan	1962	10267083	Asia	31.997	853.100710
3	Afghanistan	1967	11537966	Asia	34.020	836.197138
4	Afghanistan	1972	13079460	Asia	36.088	739.981106
...	...	...	...	...	...	...
1689	Zambia	1997	9417789	Africa	40.238	1071.353818
1690	Zambia	2002	10595811	Africa	39.193	1071.613938
1691	Zambia	2007	11746035	Africa	42.384	1271.211593
1692	Zimbabwe	1952	3080907	Africa	48.451	406.884115
1693	Zimbabwe	1957	3646340	Africa	50.469	518.764268

1694 rows × 6 columns

```
df.head()
```

	country	year	population	continent	life_exp	gdp_cap
0	Afghanistan	1952	8425333	Asia	28.801	779.445314
1	Afghanistan	1957	9240934	Asia	30.332	820.853030
2	Afghanistan	1962	10267083	Asia	31.997	853.100710
3	Afghanistan	1967	11537966	Asia	34.020	836.197138
4	Afghanistan	1972	13079460	Asia	36.088	739.981106

```
df.shape
```

```
(1704, 6)
```

```
## Create a Dataframe from the scratch
```

```
a = pd.DataFrame([["Afghanistan",   1952,   8425333,    "Asia", 28.801, 779.445314],
                ["Zimbabwe" 1952,   3080907,    "Africa"    ,48.451,    406.884115]],
                 columns=["country" "year"  "population"    continent   life_exp    gdp_cap])
```

```
  File "<ipython-input-29-bd883029badc>", line 2
    ["Zimbabwe" 1952,   3080907,        "Africa"        ,48.451,        406.884115]],
     ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

SEARCH STACK OVERFLOW

```
b = pd.DataFrame({

                  "Country":["India","USA","China"],
                  "Population":[140000000,2500000,365000]

                  })
b
```

```
df.columns
```

```
df.keys()
```

```
type(df[['country','population']])
```

```
type(df['country'])
```

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

```
df['continent'].value_counts()
```

```
# Rename the column
```

```
df.rename({ "country" : "COUNTRY","population" : "POPULATION"},axis=1,inplace=True)
```

```
df
```

```
df
```

```
## Delete a Column
```

```
df.drop('continent',axis=1)
```

```
df.drop(columns=["year"])
```

```
## Add a new column into your data frame
```

```
df["Next_decade"] = df['year']+10
df
```

```
df['gdp'] = df["POPULATION"]∗ df['gdp_cap']
df
```

```
import pandas as pd
import numpy as np
temp = pd.DataFrame([["a","b",1,3.0,]],columns=['a','b','c','d'])
temp
```

```
df
```

```
df.index.values
```

```
df.index = np.arange(1,1705, dtype="int")
```

```
df
```

```
df.index[1]
```

```
df.iloc[1]
```

```
df.loc[5]
```

```
df.iloc[[1,5,7]]
```

```
df.loc[[10,18,1056]]
```

```
df.iloc[-1]
```

```
df.iloc[0:10:2]
```

```
temp = df.set_index("continent")
temp
```

```
temp.iloc['Asia']
```

```
temp.reset_index(drop=1, inplace=1)
```

```
df.reset_index(drop=1)
```

```
df
```

```
## Add a New Row
```

```
new_row = {'country':"india",'year':2023,'population':13000000,"life_exp":56.05,'gdp_cap':678.89}
df.append(new_row,ignore_index=True)
```

```
df.loc[1705] = ["India",2025,8979807,45.78,765.90]
```

```
df
```

```
# Delete a Row from the Dataframe
```

```
df.drop([1704,1,8,10],axis=0)
```

```
df.loc[1706] = ["India",2025,8979807,"Asia",45.78,765.90]
df.loc[1707] = ["India",2025,8979807,"Asia",45.79,765.90]
df.loc[1708] = ["India",2025,8979807,"Asia",45.78,765.90]
df.loc[1709] = ["India",2025,8979807,"Asia",45.78,765.90]

df.duplicated()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-a129b84dd675> in <cell line: 1>()
----> 1 df.loc[1706] = ["India",2025,8979807,"Asia",45.78,765.90]
      2 df.loc[1707] = ["India",2025,8979807,"Asia",45.79,765.90]
      3 df.loc[1708] = ["India",2025,8979807,"Asia",45.78,765.90]
      4 df.loc[1709] = ["India",2025,8979807,"Asia",45.78,765.90]
      5

NameError: name 'df' is not defined
```

SEARCH STACK OVERFLOW

```
df[df.duplicated()]
```

```
df.loc[df.duplicated()]
```

```
df
```

```
df
```

```

df.drop_duplicates(keep=False)
```

```
## Work with Both Rows and columns
```

```
df.iloc[:4,:3]
```

```
df.loc[1:5,['country','life_exp']]
```

```
df.loc[1:5,'country':'life_exp']
```

```
df.iloc[[1,3,5],[2,4,5]]
```

```
df.loc[1:10:2,'country':'gdp_cap':2]
```

```
## Sorting
#Sorting values in either ascending order or descending order
```

```
df.sort_values(['life_exp'])
df.sort_values(['life_exp'],ascending = False)
```

```
df.sort_values(['year','life_exp'])
```

```
df.sort_values(['year','life_exp'],ascending=[True,False])
```

```
df.sort_values(['gdp_cap','population'],ascending=[False,True])
```

```
le = df["life_exp"]
le
```

```
le.min()
```

```
le.max()
```

```
le.mean()
```

```
le.count()
```

```
## Joining & Merging tables
```

```
users = pd.DataFrame({'user_id':[1,2,3,4,5],'name':['Sai','Preethi','Shamika','Veerasree','Sharan']})
users
```

```
msgs = pd.DataFrame({'user_id':[1,1,2,4],'msg':['hi','how are you?','fine','bye']})
msgs
```

```
pd.concat([users,msgs],ignore_index=True)
```

```
pd.concat([users,msgs],axis=1)
```

```
msgs
```

```
users.merge(msgs,on='user_id')
```

```
users.merge(msgs,on='user_id', how='left')
```

```
users.merge(msgs,on='user_id', how='right')
```

```
users.merge(msgs,on='user_id', how='outer')
```

```
users.rename(columns={"user_id":"id"},inplace=1)
users
```

```
users.merge(msgs,left_on='id',right_on='user_id')
```

```
!gdown 1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
```

```
!gdown 1Ws-_s1fHZ9nHfGLVUQurbHDvStePlEJm
```

```
impo

movies = pd.read_csv("movies.csv")
movies
```

```
directors = pd.read_csv("directors.csv",index_col=0)
directors
```

```
movies.shape
```

```
directors.shape
```

```
movies.ndim
```

```
directors.ndim
```

```
movies.info()
```

```
directors.info()
```

```
directors
```

```
movies.drop('Unnamed: 0', axis=1, inplace=True)
```

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

```
movies = pd.read_csv("movies.csv",index_col=0)
directors = pd.read_csv("directors.csv",index_col=0)
```

```
movies
```

```
directors
```

```
movies.head()
```

```
directors.tail()
```

```
movies['title'].nunique()
```

```
directors['id'].nunique()
```

```
movies["director_id"].nunique()
```

```
np.all(movies['director_id'].isin(directors['id']))
```

```
## Join both Movies and directors tables
```

```
data = movies.merge(directors,left_on="director_id",right_on='id',how='left')
data
```

```
data.info()
```

```
data.drop(['id_y'],axis=1,inplace=True)
```

```
data
```

```
data.info()
```

```
data.describe()
```

```
data.describe(include=object)
```

```
data['budget']= data['budget']/10000000
data
```

```
# Find out the Highly rated movies and their director details : >7
```

```
data.loc[data['vote_average']>7]
data.loc[data['vote_average']>7],['title','vote_count']
```

```
a[['title','vote_count']]
```

```
## Highly rated movies released after 2014
```

```
data.loc[(data['vote_average']>7) & (data['year']>2014)]
```

```
## Find the movies released on either Friday's or Sunday's.
```

```
data.loc[(data['day']=="Friday") | (data['day']=="Sunday")]
```

```
## Display top 10 popular movies
```

```
data.sort_values(['popularity'],ascending =False).head(10)
```

```
## Convert all males directors into 0 and Female directors into 1 in your Data Frame
```

```
def Male_Female(gender):
  if gender == "Male":
    return 0
  else:
    return 1

data['gender'] = data['gender'].apply(Male_Female)

data
```

```
## Find the Sum of Revenue and Budget
```

```
data[['revenue','budget']].sum(axis=0)
```

```
def profit(x):
  return x['revenue'] - x['budget']


data['profit'] = data[['revenue','budget']].apply(profit,axis=1)
```

```
data.sort_values('profit',ascending=False).tail()
```

```
dataa = pd.merge(movies,directors,left_on="director_id",right_on='id',how='right')
```

```
dataa
```

```
import numpy as np
import pandas as pd
!gdown 1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
!gdown 1Ws-_s1fHZ9nHfGLVUQurbHDvStePlEJm
movies = pd.read_csv("movies.csv",index_col=0)
directors = pd.read_csv("directors.csv",index_col=0)
data = pd.merge(movies,directors,left_on="director_id",right_on='id',how='left')
data.drop('id_y',axis=1,inplace=True)
data.rename({"id_x":"movies_id"},axis=1,inplace=True)
data
```

```
Downloading...
From: https://drive.google.com/uc?id=1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
To: /content/movies.csv
100% 112k/112k [00:00<00:00, 110MB/s]
Downloading...
From: https://drive.google.com/uc?id=1Ws-_s1fHZ9nHfGLVUQurbHDvStePlEJm
To: /content/directors.csv
100% 65.4k/65.4k [00:00<00:00, 108MB/s]
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	di
0	43597	237000000	150	2787965087	Avatar	7.2	11800	4762	2009	Dec	Thursday	J
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	4763	2007	May	Saturday	

```
## Grouping in Pandas

data.groupby('director_name').nunique()
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	gend
director_name												
Adam McKay	6	6	6	6	6	6	6	1	6	3	2	
Adam Shankman	8	8	7	8	8	8	8	1	7	5	2	
Alejandro González Iñárritu	6	6	6	6	6	6	6	1	6	5	3	
Alex Proyas	5	5	5	5	5	5	5	1	5	4	3	
Alexander Payne	5	5	5	5	5	3	5	1	5	4	2	
...	...	...	...	...	...	...	...	...	...	...	...	
Wes Craven	10	7	9	10	10	9	10	1	9	6	5	
Wolfgang Petersen	7	7	7	7	7	6	7	1	7	5	3	
Woody Allen	18	9	13	10	18	12	18	1	18	9	6	
Zack Snyder	7	7	7	7	7	5	7	1	7	4	4	

```
data.groupby('director_name').value_counts()
```

```
director_name  movies_id  budget     popularity  revenue    title                                 vote_average
vote_count  director_id  year  month  day        gender
Adam McKay     43882      100000000  24          170432927  The Other Guys                        6.1
1383        4925         2010  Aug    Friday     Male      1
               44151      72500000   12          162966177  Talladega Nights: The Ballad of Ricky Bobby  6.2
491         4925         2006  Aug    Friday     Male      1
               45443      26000000   29          90574188   Anchorman: The Legend of Ron Burgundy        6.7
1493        4925         2004  Jul    Friday     Male      1
               45301      28000000   57          133346506  The Big Short                         7.3
2607        4925         2015  Dec    Friday     Male      1
               44503      50000000   38          173649015  Anchorman 2: The Legend Continues     6.0
923         4925         2013  Dec    Wednesday  Male      1

..
Zhang Yimou    46460      0          21          92863945   House of Flying Daggers               7.1
439         4945         2004  May    Wednesday  Male      1
               44733      31000000   23          177394432  Hero                                  7.2
635         4945         2002  Dec    Thursday   Male      1
               44692      110        9           0          Curse of the Golden Flower            6.6
203         4945         2006  Dec    Thursday   Male      1
               43914      94000000   12          95311434   The Flowers of War                    7.1
187         4945         2011  Dec    Thursday   Male      1
               47489      0          6           0          Coming Home                           6.9
49          4945         2014  May    Friday     Male      1
Length: 1341, dtype: int64
```

```
data.groupby('director_name').ngroups
```

```
199
```

```
data.groupby('director_name').get_group('Adam Shankman')
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	di
265	44040	80000000	23	212874442	Bedtime Stories	5.9	901	4998	2008	Dec	Wednesday	Ac
300	44113	50000000	31	90450008	Hairspray	6.5	709	4998	2007	Jul	Friday	Ac
350	44195	75000000	23	59418613	Rock of Ages	6.0	385	4998	2012	Jun	Wednesday	Ac
404	44304	60000000	18	129181830	Cheaper by the Dozen 2	5.7	526	4998	2005	Dec	Wednesday	Ac

```
data.groupby('director_name').groups
```

```
{'Adam McKay': [176, 323, 366, 505, 839, 916], 'Adam Shankman': [265, 300, 350, 404, 458, 843, 999, 1231], 'Alejandro
González Iñárritu': [106, 749, 1015, 1034, 1077, 1405], 'Alex Proyas': [95, 159, 514, 671, 873], 'Alexander Payne':
[793, 1006, 1101, 1211, 1281], 'Andrew Adamson': [11, 43, 328, 501, 947], 'Andrew Niccol': [533, 603, 701, 722, 1439],
'Andrzej Bartkowiak': [349, 549, 754, 911, 924], 'Andy Fickman': [517, 681, 909, 926, 973, 1023], 'Andy Tennant': [314,
320, 464, 593, 676, 885], 'Ang Lee': [99, 134, 748, 840, 1089, 1110, 1132, 1184], 'Anne Fletcher': [610, 650, 736, 789,
1206], 'Antoine Fuqua': [310, 338, 424, 467, 576, 808, 818, 1105], 'Atom Egoyan': [946, 1128, 1164, 1194, 1347, 1416],
'Barry Levinson': [313, 319, 471, 594, 878, 898, 1013, 1037, 1082, 1143, 1185, 1345, 1378], 'Barry Sonnenfeld': [13,
48, 90, 205, 591, 778, 783], 'Ben Stiller': [209, 212, 547, 562, 850], 'Bill Condon': [102, 307, 902, 1233, 1381],
'Bobby Farrelly': [352, 356, 481, 498, 624, 630, 654, 806, 928, 972, 1111], 'Brad Anderson': [1163, 1197, 1350, 1419,
1430], 'Brett Ratner': [24, 39, 188, 207, 238, 292, 405, 456, 920], 'Brian De Palma': [228, 255, 318, 439, 747, 905,
919, 1088, 1232, 1261, 1317, 1354], 'Brian Helgeland': [512, 607, 623, 742, 933], 'Brian Levant': [418, 449, 568, 761,
860, 1003], 'Brian Robbins': [416, 441, 669, 962, 988, 1115], 'Bryan Singer': [6, 32, 33, 44, 122, 216, 297, 1326],
'Cameron Crowe': [335, 434, 488, 503, 513, 698], 'Catherine Hardwicke': [602, 695, 724, 937, 1406, 1412], 'Chris
Columbus': [117, 167, 204, 218, 229, 509, 656, 897, 996, 1086, 1129], 'Chris Weitz': [17, 500, 794, 869, 1202, 1267],
'Christopher Nolan': [3, 45, 58, 59, 74, 565, 641, 1341], 'Chuck Russell': [177, 410, 657, 1069, 1097, 1339], 'Clint
Eastwood': [369, 426, 447, 482, 490, 520, 530, 535, 645, 727, 731, 786, 787, 899, 974, 986, 1167, 1190, 1313], 'Curtis
Hanson': [494, 579, 606, 711, 733, 1057, 1310], 'Danny Boyle': [527, 668, 1083, 1085, 1126, 1168, 1287, 1385], 'Darren
Aronofsky': [113, 751, 1187, 1328, 1363, 1458], 'Darren Lynn Bousman': [1241, 1243, 1283, 1338, 1440], 'David Ayer':
[50, 273, 741, 1024, 1146, 1407], 'David Cronenberg': [541, 767, 994, 1055, 1254, 1268, 1334], 'David Fincher': [62,
213, 253, 383, 398, 478, 522, 555, 618, 785], 'David Gordon Green': [543, 862, 884, 927, 1376, 1418, 1432, 1459],
'David Koepp': [443, 644, 735, 1041, 1209], 'David Lynch': [583, 1161, 1264, 1340, 1456], 'David O. Russell': [422,
556, 609, 896, 982, 989, 1229, 1304], 'David R. Ellis': [582, 634, 756, 888, 934], 'David Zucker': [569, 619, 965,
1052, 1175], 'Dennis Dugan': [217, 260, 267, 293, 303, 718, 780, 977, 1247], 'Donald Petrie': [427, 507, 570, 649, 858,
894, 1106, 1331], 'Doug Liman': [52, 148, 251, 399, 544, 1318, 1451], 'Edward Zwick': [92, 182, 346, 566, 791, 819,
825], 'F. Gary Gray': [308, 402, 491, 523, 697, 833, 1272, 1380], 'Francis Ford Coppola': [487, 559, 622, 646, 772,
1076, 1155, 1253, 1312], 'Francis Lawrence': [63, 72, 109, 120, 679], 'Frank Coraci': [157, 249, 275, 451, 577, 599,
963], 'Frank Oz': [193, 355, 473, 580, 712, 813, 987], 'Garry Marshall': [329, 496, 528, 571, 784, 893, 1029, 1169],
'Gary Fleder': [518, 667, 689, 867, 981, 1165], 'Gary Winick': [258, 797, 798, 804, 1454], 'Gavin O'Connor': [820, 841,
939, 953, 1444], 'George A. Romero': [250, 1066, 1096, 1278, 1367, 1396], 'George Clooney': [343, 450, 831, 966, 1302],
'George Miller': [78, 103, 233, 287, 1250, 1403, 1450], 'Gore Verbinski': [1, 8, 9, 107, 119, 633, 1040], 'Guillermo
del Toro': [35, 252, 419, 486, 1118], 'Gus Van Sant': [595, 1018, 1027, 1159, 1240, 1311, 1398], 'Guy Ritchie': [124,
215, 312, 1093, 1225, 1269, 1420], 'Harold Ramis': [425, 431, 558, 586, 788, 1137, 1166, 1325], 'Ivan Reitman': [274,
643, 816, 883, 910, 935, 1134, 1242], 'James Cameron': [0, 19, 170, 173, 344, 1100, 1320], 'James Ivory': [1125, 1152,
1180, 1291, 1293, 1390, 1397], 'James Mangold': [140, 141, 557, 560, 829, 845, 958, 1145], 'James Wan': [30, 617, 1002,
1047, 1337, 1417, 1424], 'Jan de Bont': [155, 224, 231, 270, 781], 'Jason Friedberg': [812, 1010, 1012, 1014, 1036],
'Jason Reitman': [792, 1092, 1213, 1295, 1299], 'Jaume Collet-Serra': [516, 540, 640, 725, 1011, 1189], 'Jay Roach':
[195, 359, 389, 397, 461, 703, 859, 1072], 'Jean-Pierre Jeunet': [423, 485, 605, 664, 765], 'Joe Dante': [284, 525,
638, 1226, 1298, 1428], 'Joe Wright': [85, 432, 553, 803, 814, 855], 'Joel Coen': [428, 670, 691, 707, 721, 889, 906,
980, 1157, 1238, 1305], 'Joel Schumacher': [128, 184, 348, 484, 572, 614, 652, 764, 876, 886, 1108, 1230, 1280], 'John
Carpenter': [537, 663, 686, 861, 938, 1028, 1080, 1102, 1329, 1371], 'John Glen': [601, 642, 801, 847, 864], 'John
Landis': [524, 868, 1276, 1384, 1435], 'John Madden': [457, 882, 1020, 1249, 1257], 'John McTiernan': [127, 214, 244,
351, 534, 563, 648, 782, 838, 1074], 'John Singleton': [294, 489, 732, 796, 1120, 1173, 1316], 'John Whitesell': [499,
632, 763, 1119, 1148], 'John Woo': [131, 142, 264, 371, 420, 675, 1182], 'Jon Favreau': [46, 54, 55, 382, 759, 1346],
'Jon M. Chu': [100, 225, 810, 1099, 1186], 'Jon Turteltaub': [64, 180, 372, 480, 760, 846, 1171], 'Jonathan Demme':
[277, 493, 1000, 1123, 1215], 'Jonathan Liebesman': [81, 143, 339, 1117, 1301], 'Judd Apatow': [321, 710, 717, 865,
881], 'Justin Lin': [38, 123, 246, 1437, 1447], 'Kenneth Branagh': [80, 197, 421, 879, 1094, 1277, 1288], 'Kenny
Ortega': [412, 852, 1228, 1315, 1365], 'Kevin Reynolds': [53, 502, 639, 1019, 1059], ...}
```

```
data.groupby('director_name')['title'].count().sort_values(ascending=False)
```

```
director_name
Steven Spielberg    26
Clint Eastwood      19
Martin Scorsese     19
Woody Allen         18
Robert Rodriguez    16
                    ..
Paul Weitz           5
John Madden          5
Paul Verhoeven       5
John Whitesell       5
Kevin Reynolds       5
Name: title, Length: 199, dtype: int64
```

```
data.groupby('director_name')['year'].aggregate(['min','max'])
```

	min	max
director_name		
Adam McKay	2004	2015
Adam Shankman	2001	2012
Alejandro González Iñárritu	2000	2015
Alex Proyas	1994	2016
Alexander Payne	1999	2013
...	...	...
Wes Craven	1984	2011
Wolfgang Petersen	1981	2006
Woody Allen	1977	2013

```
## Get me the list of High budget directors
## - At least 1 movie with 100 Million budget.


data_dir_budget = data.groupby('director_name')['budget'].max().reset_index()
data_dir_budget
```

	director_name	budget
0	Adam McKay	100000000
1	Adam Shankman	80000000
2	Alejandro González Iñárritu	135000000
3	Alex Proyas	140000000
4	Alexander Payne	30000000
...	...	...
194	Wes Craven	40000000
195	Wolfgang Petersen	175000000
196	Woody Allen	30000000
197	Zack Snyder	250000000
198	Zhang Yimou	94000000

199 rows × 2 columns

```
names = data_dir_budget[data_dir_budget['budget']>=100000000]['director_name']
```

```
data.loc[data['director_name'].isin(names)]
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	di
0	43597	237000000	150	2787965087	Avatar	7.2	11800	4762	2009	Dec	Thursday	J
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	4763	2007	May	Saturday	
2	43599	245000000	107	880674609	Spectre	6.3	4466	4764	2015	Oct	Monday	
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106	4765	2012	Jul	Monday	
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576	4767	2007	May	Tuesday	
...	...	...	...	...	...	...	...	...	...	...	...	
1450	48267	400000	33	100000000	Mad Max	6.6	1213	4845	1979	Apr	Thursday	
1451	48268	200000	13	4505922	Swingers	6.8	253	4813	1996	Oct	Friday	

```
def high_budget(data):
  return data['budget'].max()>=100000000

data.groupby('director_name').filter(high_budget)
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	di
0	43597	237000000	150	2787965087	Avatar	7.2	11800	4762	2009	Dec	Thursday	J
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	4763	2007	May	Saturday	
2	43599	245000000	107	880674609	Spectre	6.3	4466	4764	2015	Oct	Monday	
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106	4765	2012	Jul	Monday	
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576	4767	2007	May	Tuesday	
...	...	...	...	...	...	...	...	...	...	...	...	
1450	48267	400000	33	100000000	Mad Max	6.6	1213	4845	1979	Apr	Thursday	
1451	48268	200000	13	4505922	Swingers	6.8	253	4813	1996	Oct	Friday	
1452	[illegible]	0	5	[illegible]	Th	[illegible]	[illegible]	[illegible]	[illegible]	D	Th	

```
## Find out the Risky Movies!

# Average Revenue of the Director - 10,20,15,20,18 - 21M
# Risky - >21M  : 25M,30M,18M, 10M, 50M
```

```
def is_risky(x):
  x['is_risky'] = (x['budget']-x['revenue'].mean())<0
  return x

data_risky = data.groupby('director_name').apply(is_risky)
data_risky
```

```
<ipython-input-44-06b5bf5a898e>:4: FutureWarning: Not prepending group keys to the result index of transform-like apply.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  data_risky = data.groupby('director_name').apply(is_risky)
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	di
0	43597	237000000	150	2787965087	Avatar	7.2	11800	4762	2009	Dec	Thursday	J
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	4763	2007	May	Saturday	
2	43599	245000000	107	880674609	Spectre	6.3	4466	4764	2015	Oct	Monday	
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106	4765	2012	Jul	Monday	
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576	4767	2007	May	Tuesday	
...	...	...	...	...	...	...	...	...	...	...	...	
1460	48363	0	3	321952	The Last Waltz	7.9	64	4809	1978	May	Monday	M
1461	48370	27000	19	3151130	Clerks	7.4	755	5369	1994	Sep	Tuesday	
1462	48375	0	7	0	Rampage	6.0	131	5148	2009	Aug	Friday	

```
def is_risky(x):
  x['is_risky'] = (x['budget']-x['revenue'].mean())<0
  return x
data_risky = data.groupby('director_name').apply(is_risky)
data_risky.loc[data_risky['is_risky']==True]
```

```
<ipython-input-45-7e0ace731308>:4: FutureWarning: Not prepending group keys to the result index of transform-like apply.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  data_risky = data.groupby('director_name').apply(is_risky)
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	di
0	43597	237000000	150	2787965087	Avatar	7.2	11800	4762	2009	Dec	Thursday	J
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	4763	2007	May	Saturday	
2	43599	245000000	107	880674609	Spectre	6.3	4466	4764	2015	Oct	Monday	
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106	4765	2012	Jul	Monday	
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576	4767	2007	May	Tuesday	
...	...	...	...	...	...	...	...	...	...	...	...	
1460	48363	0	3	321952	The Last Waltz	7.9	64	4809	1978	May	Monday	M

```
## Multi indexing
```

```
import numpy as np
import pandas as pd
!gdown 1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
!gdown 1Ws-_s1fHZ9nHfGLVUQurbHDvStePlEJm
movies = pd.read_csv("movies.csv",index_col=0)
directors = pd.read_csv("directors.csv",index_col=0)
data = pd.merge(movies,directors,left_on="director_id",right_on='id',how='left')
data.drop('id_y',axis=1,inplace=True)
data.rename({"id_x":"movies_id"},axis=1,inplace=True)
data
```

```
Downloading...
From: https://drive.google.com/uc?id=1s2TkjSpzNc4SyxqRrQleZyDIHlc7bxnd
To: /content/movies.csv
100% 112k/112k [00:00<00:00, 46.8MB/s]
Downloading...
From: https://drive.google.com/uc?id=1Ws-_s1fHZ9nHfGLVUQurbHDvStePlEJm
To: /content/directors.csv
100% 65.4k/65.4k [00:00<00:00, 77.1MB/s]
```

	movies_id	budget	popularity	revenue	title	vote_average	vote_count	director_id	year	month	day	di
0	43597	237000000	150	2787965087	Avatar	7.2	11800	4762	2009	Dec	Thursday	J
1	43598	300000000	139	961000000	Pirates of the Caribbean: At World's End	6.9	4500	4763	2007	May	Saturday	
2	43599	245000000	107	880674609	Spectre	6.3	4466	4764	2015	Oct	Monday	
3	43600	250000000	112	1084939099	The Dark Knight Rises	7.6	9106	4765	2012	Jul	Monday	
4	43602	258000000	115	890871626	Spider-Man 3	5.9	3576	4767	2007	May	Tuesday	
...	...	...	...	...	...	...	...	...	...	...	...	
1460	48363	0	3	321952	The Last Waltz	7.9	64	4809	1978	May	Monday	M
1461	48370	27000	19	3151130	Clerks	7.4	755	5369	1994	Sep	Tuesday	

```
data_agg = data.groupby(['director_name'])[['title','year']].aggregate({'title':'count','year':['min','max']})
```

```
data.columns
```

```
Index(['movies_id', 'budget', 'popularity', 'revenue', 'title', 'vote_average',
       'vote_count', 'director_id', 'year', 'month', 'day', 'director_name',
       'gender'],
      dtype='object')
```

```
data_agg.columns
```

```
MultiIndex([('title', 'count'),
            ( 'year',   'min'),
            ( 'year',   'max')],
           )
```

```
data_agg
```

	title	year	
	count	min	max
director_name			
Adam McKay	6	2004	2015
Adam Shankman	8	2001	2012
Alejandro González Iñárritu	6	2000	2015
Alex Proyas	5	1994	2016
Alexander Payne	5	1999	2013
...	...	...	...
Wes Craven	10	1984	2011
Wolfgang Petersen	7	1981	2006
Woody Allen	18	1977	2013
Zack Snyder	7	2004	2016
Zhang Yimou	6	2002	2014

199 rows × 3 columns

```
data_agg.columns = ['_'.join(tup) for tup in data_agg.columns]
data_agg
```

	title_count	year_min	year_max
director_name			
Adam McKay	6	2004	2015
Adam Shankman	8	2001	2012
Alejandro González Iñárritu	6	2000	2015
Alex Proyas	5	1994	2016
Alexander Payne	5	1999	2013
...	...	...	...
Wes Craven	10	1984	2011
Wolfgang Petersen	7	1981	2006
Woody Allen	18	1977	2013
Zack Snyder	7	2004	2016
Zhang Yimou	6	2002	2014

199 rows × 3 columns

```
!gdown 173A59xh2mnpmljCCB9bhC4C5eP2IS6qZ
```

```
Downloading...
From: https://drive.google.com/uc?id=173A59xh2mnpmljCCB9bhC4C5eP2IS6qZ
To: /content/Pfizer_1.csv
100% 1.51k/1.51k [00:00<00:00, 7.10MB/s]
```

```
data = pd.read_csv('Pfizer_1.csv')
data
```

	Date	Drug_Name	Parameter	1:30:00	2:30:00	3:30:00	4:30:00	5:30:00	6:30:00	7:30:00	8:30:00	9:30:00	10:30:00
0	15-10-2020	diltiazem hydrochloride	Temperature	23.0	22.0	NaN	21.0	21.0	22	23.0	21.0	22.0	20
1	15-10-2020	diltiazem hydrochloride	Pressure	12.0	13.0	NaN	11.0	13.0	14	16.0	16.0	24.0	18
2	15-10-2020	docetaxel injection	Temperature	NaN	17.0	18.0	NaN	17.0	18	NaN	NaN	23.0	23
3	15-10-2020	docetaxel injection	Pressure	NaN	22.0	22.0	NaN	22.0	23	NaN	NaN	27.0	26
4	15-10-2020	ketamine hydrochloride	Temperature	24.0	NaN	NaN	27.0	NaN	26	25.0	24.0	23.0	22
5	15-10-2020	ketamine hydrochloride	Pressure	8.0	NaN	NaN	7.0	NaN	9	10.0	11.0	10.0	9
6	16-10-2020	diltiazem hydrochloride	Temperature	34.0	35.0	36.0	36.0	37.0	38	37.0	38.0	39.0	40

```
data.shape
```

```
(18, 15)
```

docetaxel

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 15 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       18 non-null     object
 1   Drug_Name  18 non-null     object
 2   Parameter  18 non-null     object
 3   1:30:00    16 non-null     float64
 4   2:30:00    16 non-null     float64
 5   3:30:00    12 non-null     float64
 6   4:30:00    14 non-null     float64
 7   5:30:00    16 non-null     float64
 8   6:30:00    18 non-null     int64
 9   7:30:00    16 non-null     float64
 10  8:30:00    14 non-null     float64
 11  9:30:00    16 non-null     float64
 12  10:30:00   18 non-null     int64
 13  11:30:00   16 non-null     float64
 14  12:30:00   18 non-null     int64
dtypes: float64(9), int64(3), object(3)
memory usage: 2.2+ KB
```

```
pd.melt(data,id_vars=['Date','Drug_Name','Parameter'])
```

	Date	Drug_Name	Parameter	variable	value
0	15-10-2020	diltiazem hydrochloride	Temperature	1:30:00	23.0
1	15-10-2020	diltiazem hydrochloride	Pressure	1:30:00	12.0
2	15-10-2020	docetaxel injection	Temperature	1:30:00	NaN
3	15-10-2020	docetaxel injection	Pressure	1:30:00	NaN
4	15-10-2020	ketamine hydrochloride	Temperature	1:30:00	24.0
...	...	...	...	...	...
211	17-10-2020	diltiazem hydrochloride	Pressure	12:30:00	14.0
212	17-10-2020	docetaxel injection	Temperature	12:30:00	23.0
213	17-10-2020	docetaxel injection	Pressure	12:30:00	28.0
214	17-10-2020	ketamine hydrochloride	Temperature	12:30:00	24.0
215	17-10-2020	ketamine hydrochloride	Pressure	12:30:00	15.0

216 rows × 5 columns

```
pd.melt?
```

```
data_melt = pd.melt(data,id_vars=['Date','Drug_Name','Parameter'],var_name='Time',value_name='Reading')
data_melt
```

	Date	Drug_Name	Parameter	Time	Reading
0	15-10-2020	diltiazem hydrochloride	Temperature	1:30:00	23.0
1	15-10-2020	diltiazem hydrochloride	Pressure	1:30:00	12.0
2	15-10-2020	docetaxel injection	Temperature	1:30:00	NaN
3	15-10-2020	docetaxel injection	Pressure	1:30:00	NaN
4	15-10-2020	ketamine hydrochloride	Temperature	1:30:00	24.0
...	...	...	...	...	...
211	17-10-2020	diltiazem hydrochloride	Pressure	12:30:00	14.0
212	17-10-2020	docetaxel injection	Temperature	12:30:00	23.0
213	17-10-2020	docetaxel injection	Pressure	12:30:00	28.0
214	17-10-2020	ketamine hydrochloride	Temperature	12:30:00	24.0
215	17-10-2020	ketamine hydrochloride	Pressure	12:30:00	15.0

216 rows × 5 columns

```
data_melt.shape
```

```
(216, 5)
```

```
data_melt.pivot?
```

```
data_melt.pivot(index=['Date',"Drug_Name","Parameter"],columns="Time",values="Reading").reset_index()
```

Time	Date	Drug_Name	Parameter	10:30:00	11:30:00	12:30:00	1:30:00	2:30:00	3:30:00	4:30:00	5:30:00	6:30:00	7:30
0	15-10-2020	diltiazem hydrochloride	Pressure	18.0	19.0	20.0	12.0	13.0	NaN	11.0	13.0	14.0	1
1	15-10-2020	diltiazem hydrochloride	Temperature	20.0	20.0	21.0	23.0	22.0	NaN	21.0	21.0	22.0	2
2	15-10-2020	docetaxel injection	Pressure	26.0	29.0	28.0	NaN	22.0	22.0	NaN	22.0	23.0	N
3	15-10-2020	docetaxel injection	Temperature	23.0	25.0	25.0	NaN	17.0	18.0	NaN	17.0	18.0	N
4	15-10-2020	ketamine hydrochloride	Pressure	9.0	9.0	11.0	8.0	NaN	NaN	7.0	NaN	9.0	1
5	15-10-2020	ketamine hydrochloride	Temperature	22.0	21.0	20.0	24.0	NaN	NaN	27.0	NaN	26.0	2
6	16-10-2020	diltiazem hydrochloride	Pressure	24.0	NaN	27.0	18.0	19.0	20.0	21.0	22.0	23.0	2
7	16-10-2020	diltiazem hydrochloride	Temperature	40.0	NaN	42.0	34.0	35.0	36.0	36.0	37.0	38.0	3
8	16-10-	docetaxel injection	Pressure	28.0	29.0	30.0	23.0	24.0	NaN	25.0	26.0	27.0	2

```
data_melt
```

	Date	Drug_Name	Parameter	Time	Reading
0	15-10-2020	diltiazem hydrochloride	Temperature	1:30:00	23.0
1	15-10-2020	diltiazem hydrochloride	Pressure	1:30:00	12.0
2	15-10-2020	docetaxel injection	Temperature	1:30:00	NaN
3	15-10-2020	docetaxel injection	Pressure	1:30:00	NaN
4	15-10-2020	ketamine hydrochloride	Temperature	1:30:00	24.0
...	...	...	...	...	...
211	17-10-2020	diltiazem hydrochloride	Pressure	12:30:00	14.0
213	17-10-2020	docetaxel injection	Pressure	12:30:00	28.0

```
data_tidy = data_melt.pivot(index=['Date','Drug_Name',"Time"],columns='Parameter',values='Reading').reset_index()
```

```
data_tidy
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.0
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.0
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.0
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.0
...	...	...	...	...	...
103	17-10-2020	ketamine hydrochloride	5:30:00	11.0	17.0
104	17-10-2020	ketamine hydrochloride	6:30:00	12.0	18.0
105	17-10-2020	ketamine hydrochloride	7:30:00	12.0	19.0
106	17-10-2020	ketamine hydrochloride	8:30:00	11.0	20.0
107	17-10-2020	ketamine hydrochloride	9:30:00	12.0	21.0

108 rows × 5 columns

```
type(None)
```

```
NoneType
```

```
type(np.nan)
```

```
float
```

```
a = pd.Series([1,np.nan,3])
type(a[1])
```

```
numpy.float64
```

```
b = pd.Series(['1','np.nan','3',None])
b
```

```
0         1
1    np.nan
2         3
3      None
dtype: object
```

```
pd.Series([1,2,3,np.nan])
```

```
0    1.0
1    2.0
2    3.0
3    NaN
dtype: float64
```

```
## How to deal with null values
```

```
data.isnull().sum(axis=1)
```

```
0     1
1     1
2     4
3     4
```

```
4     3
5     3
6     1
7     1
8     1
9     1
10    2
11    2
12    1
13    1
14    0
15    0
16    0
17    0
dtype: int64
```

```
data.isnull().sum()
```

```
Date          0
Drug_Name     0
Parameter     0
1:30:00       2
2:30:00       2
3:30:00       6
4:30:00       4
5:30:00       2
6:30:00       0
7:30:00       2
8:30:00       4
9:30:00       2
10:30:00      0
11:30:00      2
12:30:00      0
dtype: int64
```

```
data.shape
```

```
(18, 15)
```

```
data.dropna()
```

	Date	Drug_Name	Parameter	1:30:00	2:30:00	3:30:00	4:30:00	5:30:00	6:30:00	7:30:00	8:30:00	9:30:00	10:30:00
14	17-10-2020	docetaxel injection	Temperature	12.0	13.0	14.0	15.0	16.0	17	18.0	19.0	20.0	21
15	17-10-2020	docetaxel injection	Pressure	20.0	22.0	22.0	22.0	22.0	23	25.0	26.0	27.0	28

```
data.fillna(0)
```

	Date	Drug_Name	Parameter	1:30:00	2:30:00	3:30:00	4:30:00	5:30:00	6:30:00	7:30:00	8:30:00	9:30:00	10:30:00

```
data['2:30:00'].fillna(data['2:30:00'].mean())
```

```
0     22.0000
1     13.0000
2     17.0000
3     22.0000
4     18.8125
5     18.8125
6     35.0000
7     19.0000
8     47.0000
9     24.0000
10     9.0000
11    12.0000
12    19.0000
13     4.0000
14    13.0000
15    22.0000
16    14.0000
17     9.0000
Name: 2:30:00, dtype: float64
```

```
data['2:30:00']
```

```
0     22.0
1     13.0
2     17.0
3     22.0
4      NaN
5      NaN
6     35.0
7     19.0
8     47.0
9     24.0
10     9.0
11    12.0
12    19.0
13     4.0
14    13.0
15    22.0
16    14.0
17     9.0
Name: 2:30:00, dtype: float64
```

```
data[:20]
```

	Date	Drug_Name	Parameter	1:30:00	2:30:00	3:30:00	4:30:00	5:30:00	6:30:00	7:30:00	8:30:00	9:30:00	10:30:00
0	15-10-2020	diltiazem hydrochloride	Temperature	23.0	22.0	NaN	21.0	21.0	22	23.0	21.0	22.0	20
1	15-10-2020	diltiazem hydrochloride	Pressure	12.0	13.0	NaN	11.0	13.0	14	16.0	16.0	24.0	18
2	15-10-2020	docetaxel injection	Temperature	NaN	17.0	18.0	NaN	17.0	18	NaN	NaN	23.0	23
3	15-10-2020	docetaxel injection	Pressure	NaN	22.0	22.0	NaN	22.0	23	NaN	NaN	27.0	26
4	15-10-2020	ketamine hydrochloride	Temperature	24.0	NaN	NaN	27.0	NaN	26	25.0	24.0	23.0	22
5	15-10-2020	ketamine hydrochloride	Pressure	8.0	NaN	NaN	7.0	NaN	9	10.0	11.0	10.0	9
6	16-10-2020	diltiazem hydrochloride	Temperature	34.0	35.0	36.0	36.0	37.0	38	37.0	38.0	39.0	40
7	16-10-2020	diltiazem hydrochloride	Pressure	18.0	19.0	20.0	21.0	22.0	23	24.0	25.0	25.0	24
8	16-10-	docetaxel injection	Temperature	46.0	47.0	NaN	48.0	48.0	49	50.0	52.0	55.0	56

```
data_tidy
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.0
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.0
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.0
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.0
...	...	...	...	...	...
103	17-10-2020	ketamine hydrochloride	5:30:00	11.0	17.0
104	17-10-2020	ketamine hydrochloride	6:30:00	12.0	18.0
105	17-10-2020	ketamine hydrochloride	7:30:00	12.0	19.0
106	17-10-2020	ketamine hydrochloride	8:30:00	11.0	20.0
107	17-10-2020	ketamine hydrochloride	9:30:00	12.0	21.0

108 rows × 5 columns

```
def temp_mean(x):
  x['temp_avg'] = x['Temperature'].mean()
  return x

data_tidy = data_tidy.groupby('Drug_Name').apply(temp_mean)
data_tidy
```

```
<ipython-input-38-dbe14b0a63e9>:5: FutureWarning: Not prepending group keys to the result index of transform-like apply.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  data_tidy = data_tidy.groupby('Drug_Name').apply(temp_mean)
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0	24.848485
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.0	24.848485
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.0	24.848485
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.0	24.848485
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.0	24.848485
...	...	...	...	...	...	...
103	17-10-2020	ketamine hydrochloride	5:30:00	11.0	17.0	17.709677
104	17-10-2020	ketamine hydrochloride	6:30:00	12.0	18.0	17.709677
105	17-10-2020	ketamine hydrochloride	7:30:00	12.0	19.0	17.709677
106	17-10-2020	ketamine hydrochloride	8:30:00	11.0	20.0	17.709677
107	17-10-2020	ketamine hydrochloride	9:30:00	12.0	21.0	17.709677

108 rows × 6 columns

```
data_tidy[:20]
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0	24.848485
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.0	24.848485
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.0	24.848485
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.0	24.848485
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.0	24.848485
5	15-10-2020	diltiazem hydrochloride	3:30:00	NaN	NaN	24.848485
6	15-10-2020	diltiazem hydrochloride	4:30:00	11.0	21.0	24.848485
7	15-10-2020	diltiazem hydrochloride	5:30:00	13.0	21.0	24.848485
8	15-10-2020	diltiazem hydrochloride	6:30:00	14.0	22.0	24.848485
9	15-10-2020	diltiazem hydrochloride	7:30:00	16.0	23.0	24.848485

```
data_tidy['Temperature'].fillna(data_tidy['temp_avg'],inplace=True)
data_tidy[:20]
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.000000	24.848485
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.000000	24.848485
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.000000	24.848485
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.000000	24.848485
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.000000	24.848485
5	15-10-2020	diltiazem hydrochloride	3:30:00	NaN	24.848485	24.848485
6	15-10-2020	diltiazem hydrochloride	4:30:00	11.0	21.000000	24.848485
7	15-10-2020	diltiazem hydrochloride	5:30:00	13.0	21.000000	24.848485
8	15-10-2020	diltiazem hydrochloride	6:30:00	14.0	22.000000	24.848485
9	15-10-2020	diltiazem hydrochloride	7:30:00	16.0	23.000000	24.848485
10	15-10-2020	diltiazem hydrochloride	8:30:00	16.0	21.000000	24.848485
11	15-10-2020	diltiazem hydrochloride	9:30:00	24.0	22.000000	24.848485
12	15-10-2020	docetaxel injection	10:30:00	26.0	23.000000	30.387097
13	15-10-2020	docetaxel injection	11:30:00	29.0	25.000000	30.387097
14	15-10-2020	docetaxel injection	12:30:00	28.0	25.000000	30.387097
15	15-10-2020	docetaxel injection	1:30:00	NaN	30.387097	30.387097
16	15-10-2020	docetaxel injection	2:30:00	22.0	17.000000	30.387097
17	15-10-2020	docetaxel injection	3:30:00	22.0	18.000000	30.387097
18	15-10-2020	docetaxel injection	4:30:00	NaN	30.387097	30.387097
19	15-10-2020	docetaxel injection	5:30:00	22.0	17.000000	30.387097

```
def p_m(x):
  x['pres_avg'] = x['Pressure'].mean()
  return x

data_tidy = data_tidy.groupby('Drug_Name').apply(p_m)
data_tidy

data_tidy['Pressure'].fillna(data_tidy['pres_avg'],inplace=True)
data_tidy[:20]
```

```
<ipython-input-41-b0131e92ccef>:5: FutureWarning: Not prepending group keys to the result index of transform-like apply.
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  data_tidy = data_tidy.groupby('Drug_Name').apply(p_m)
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg	pres_avg
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.000000	20.000000	24.848485	15.424242
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.000000	20.000000	24.848485	15.424242
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.000000	21.000000	24.848485	15.424242
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.000000	23.000000	24.848485	15.424242
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.000000	22.000000	24.848485	15.424242
5	15-10-2020	diltiazem hydrochloride	3:30:00	15.424242	24.848485	24.848485	15.424242
6	15-10-2020	diltiazem hydrochloride	4:30:00	11.000000	21.000000	24.848485	15.424242
7	15-10-2020	diltiazem hydrochloride	5:30:00	13.000000	21.000000	24.848485	15.424242
8	15-10-2020	diltiazem hydrochloride	6:30:00	14.000000	22.000000	24.848485	15.424242
9	15-10-2020	diltiazem hydrochloride	7:30:00	16.000000	23.000000	24.848485	15.424242
10	15-10-2020	diltiazem hydrochloride	8:30:00	16.000000	21.000000	24.848485	15.424242
11	15-10-2020	diltiazem hydrochloride	9:30:00	24.000000	22.000000	24.848485	15.424242

```
## Binning the data
```

13 15-10-2020 docetaxel injection 11:30:00 29.000000 25.000000 30.387097 25.483871

```
data_tidy['Temperature'].min()
```

```
8.0
```

16 15-10-2020 docetaxel injection 2:30:00 22.000000 17.000000 30.387097 25.483871

```
data_tidy['Temperature'].max()
```

```
58.0
```

```
data_tidy['Pressure'].min()
```

```
3.0
```

```
data_tidy['Pressure'].max()
```

```
30.0
```

```
temp_points = [ 5,20,35,50,60]
temp_names = ['low','medium','high','very_high']

data_tidy['Temp_category'] = pd.cut(data_tidy['Temperature'],bins =temp_points,labels=temp_names)
data_tidy
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg	pres_avg	Temp_category
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0	24.848485	15.424242	low
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.0	24.848485	15.424242	low
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.0	24.848485	15.424242	medium
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.0	24.848485	15.424242	medium
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.0	24.848485	15.424242	medium
...	...	...	...	...	...	...	...	...
103	17-10-2020	ketamine hydrochloride	5:30:00	11.0	17.0	17.709677	11.935484	low
104	17-10-2020	ketamine hydrochloride	6:30:00	12.0	18.0	17.709677	11.935484	low
105	17-10-2020	ketamine hydrochloride	7:30:00	12.0	19.0	17.709677	11.935484	low
106	17-10-2020	ketamine hydrochloride	8:30:00	11.0	20.0	17.709677	11.935484	low
107	17-10-2020	ketamine hydrochloride	9:30:00	12.0	21.0	17.709677	11.935484	medium

108 rows × 8 columns

```
pres_points = [0,18,25,35]
pres_names = ['Below_average','Average','Above_average']

data_tidy['Pres_category'] = pd.cut(data_tidy['Pressure'],bins =pres_points,labels=pres_names)
data_tidy
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg	pres_avg	Temp_category	Pres_category
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0	24.848485	15.424242	low	Below_average
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.0	24.848485	15.424242	low	Average
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.0	24.848485	15.424242	medium	Average
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.0	24.848485	15.424242	medium	Below_average
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.0	24.848485	15.424242	medium	Below_average
...	...	...	...	...	...	...	...	...	...
103	17-10-2020	ketamine hydrochloride	5:30:00	11.0	17.0	17.709677	11.935484	low	Below_average
104	17-10-2020	ketamine hydrochloride	6:30:00	12.0	18.0	17.709677	11.935484	low	Below_average

```
data_tidy['Temp_category'].value_counts()
```

```
low          50
medium       38
high         15
very_high     5
Name: Temp_category, dtype: int64
```

```
data_tidy['Pres_category'].value_counts()
```

```
Below_average    59
Average          26
Above_average    23
Name: Pres_category, dtype: int64
```

```
data_tidy.loc[data_tidy["Drug_Name"]=="hydrochloride"]
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg	pres_avg	Temp_category	Pres_category

```
data_tidy.loc[data_tidy["Drug_Name"].str.contains('hydrochloride',case=0)]
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg	pres_avg	Temp_category	Pres_category
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0	24.848485	15.424242	low	Below_average
1	15-10-2020	diltiazem hydrochloride	11:30:00	19.0	20.0	24.848485	15.424242	low	Average
2	15-10-2020	diltiazem hydrochloride	12:30:00	20.0	21.0	24.848485	15.424242	medium	Average
3	15-10-2020	diltiazem hydrochloride	1:30:00	12.0	23.0	24.848485	15.424242	medium	Below_average
4	15-10-2020	diltiazem hydrochloride	2:30:00	13.0	22.0	24.848485	15.424242	medium	Below_average
...	...	...	...	...	...	...	...	...	...
103	17-10-2020	ketamine hydrochloride	5:30:00	11.0	17.0	17.709677	11.935484	low	Below_average
104	17-10-2020	ketamine hydrochloride	6:30:00	12.0	18.0	17.709677	11.935484	low	Below_average

DATE And Time Functions

```
data_tidy[['Date','Time']]
```

Parameter	Date	Time
0	15-10-2020	10:30:00
1	15-10-2020	11:30:00
2	15-10-2020	12:30:00
3	15-10-2020	1:30:00
4	15-10-2020	2:30:00
...	...	...
103	17-10-2020	5:30:00
104	17-10-2020	6:30:00
105	17-10-2020	7:30:00
106	17-10-2020	8:30:00
107	17-10-2020	9:30:00

108 rows × 2 columns

```
type(data_tidy['Date'])
```

```
pandas.core.series.Series
```

```
def get_year(Date):
  return Date[2]

data_tidy['Date'].str.split('-').apply(get_year)
```

```
0      2020
1      2020
2      2020
3      2020
4      2020
       ...
103    2020
104    2020
105    2020
106    2020
107    2020
Name: Date, Length: 108, dtype: object
```

```
def abc(Date):
  return Date[0]

data_tidy['Date'].str.split('/').apply(abc)
```

```
0      15-10-2020
1      15-10-2020
2      15-10-2020
3      15-10-2020
4      15-10-2020
          ...
103    17-10-2020
104    17-10-2020
105    17-10-2020
106    17-10-2020
107    17-10-2020
Name: Date, Length: 108, dtype: object
```

```
data_tidy['time_stamp'] = data_tidy['Date'] + " " + data_tidy['Time']
data_tidy
```

Parameter	Date	Drug_Name	Time	Pressure	Temperature	temp_avg	pres_avg	Temp_category	Pres_category	time_stamp
0	15-10-2020	diltiazem hydrochloride	10:30:00	18.0	20.0	24.848485	15.424242	low	Below_average	15-10-2020 10:30:00
	15-	diltiazem								15-10-2020
	2020	hydrochloride								1:30:00
...	...	...	...	...	...	...	...	...	...	...

```
type(data_tidy['time_stamp'])
```

```
pandas.core.series.Series
```

```
## Converting String date into date format
```

```
data_tidy['time_stamp'] = pd.to_datetime(data_tidy['time_stamp'])
type(data_tidy['time_stamp'][1])
```

```
pandas._libs.tslibs.timestamps.Timestamp
```

```
Date = data_tidy['time_stamp'][9]
```

```
Date.year
```

```
2020
```

```
Date.month
```

```
10
```

```
Date.day
```

```
15
```

```
Date.minute
```

```
30
```

```
Date.month_name()
```

```
'October'
```

```
data_tidy['time_stamp'].dt.day_name()
```

```
0      Thursday
1      Thursday
2      Thursday
3      Thursday
4      Thursday
         ...
103    Saturday
104    Saturday
105    Saturday
106    Saturday
107    Saturday
Name: time_stamp, Length: 108, dtype: object
```

```
data_tidy['time_stamp'][9]
```

```
Timestamp('2020-10-15 07:30:00')
```

```
Date
```

```
Timestamp('2020-10-15 10:30:00')
```

```
Date.strftime('%y')
```

```
'20'
```

```
Date.strftime('%d')
```

```
'15'
```

```
Date.strftime('%d-%m-%y')
```

```
    '15-10-20'
```

```
Date.strftime('%m-%h')
```

```
    '10-Oct'
```

```
### Save the File
```

```
data_tidy.to_csv('Pfizer_tidy.csv',sep=",")
```

```
data_tidy.to_excel('Pfizer_tidy.xlsx')
```