

Pandas 資料分析 (3)

張家瑋 副教授

國立臺中科技大學資訊工程系

建立 DataFrame

```
import pandas as pd
import numpy as np
pd.set_option('max_columns', 4, 'max_rows', 10, 'max_colwidth', 12)
```

```
fname = ['Paul', 'John', 'Richard', 'George']
lname = ['McCartney', 'Lennon', 'Starkey', 'Harrison']
birth = [1942, 1940, 1940, 1943]
```

```
people = {'first': fname, 'last': lname, 'birth': birth}
```

```
beatles = pd.DataFrame(people)
beatles
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

```
beatles.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

```
beatles.columns
```

```
Index(['first', 'last', 'birth'], dtype='object')
```

```
pd.DataFrame(people, index=['a', 'b', 'c', 'd'])
```

	first	last	birth
a	Paul	McCartney	1942
b	John	Lennon	1940
c	Richard	Starkey	1940
d	George	Harrison	1943

建立 DataFrame

```
pd.DataFrame([{"first": "Paul", "last": "McCartney", "birth": 1942},  
              {"first": "John", "last": "Lennon", "birth": 1940},  
              {"first": "Richard", "last": "Starkey", "birth": 1940},  
              {"first": "George", "last": "Harrison", "birth": 1943}])
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

```
pd.DataFrame([{"first": "Paul", "last": "McCartney", "birth": 1942},  
              {"first": "John", "last": "Lennon", "birth": 1940},  
              {"first": "Richard", "last": "Starkey", "birth": 1940},  
              {"first": "George", "last": "Harrison", "birth": 1943}],  
             columns=['last', 'first', 'birth'])
```

	last	first	birth
0	McCartney	Paul	1942
1	Lennon	John	1940
2	Starkey	Richard	1940
3	Harrison	George	1943

CSV 建立 DataFrame

beatles

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

```
from io import StringIO
fout = StringIO()
beatles.to_csv(fout)
```

```
print(fout.getvalue())
```

```
,first,last,birth
0,Paul,McCartney,1942
1,John,Lennon,1940
2,Richard,Starkey,1940
3,George,Harrison,1943
```

```
fout.seek(0)
pd.read_csv(fout)
```

	Unnamed: 0	first	last	birth
0	0	Paul	McCartney	1942
1	1	John	Lennon	1940
2	2	Richard	Starkey	1940
3	3	George	Harrison	1943

```
fout.seek(0)
pd.read_csv(fout, index_col=0)
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

```
fout = StringIO()
beatles.to_csv(fout, index=False)
print(fout.getvalue())
```

```
first,last,birth
Paul,McCartney,1942
John,Lennon,1940
Richard,Starkey,1940
George,Harrison,1943
```

由於 csv 的索引無法直接當成 dataframe 的索引，因此使用 seek(0) 建立新索引。

原本的 csv 索引就變成了 Unnamed:0。

在 read_csv() 中，直接指定 index_col = 0，即特定(第一個)欄位作為 dataframe 索引。

讀取大型 CSV 檔案

```
diamonds = pd.read_csv('data/diamonds.csv', nrows=1000)
diamonds
```

	carat	cut	...	y	z
0	0.23	Ideal	...	3.98	2.43
1	0.21	Premium	...	3.84	2.31
2	0.23	Good	...	4.07	2.31
3	0.29	Premium	...	4.23	2.63
4	0.31	Good	...	4.35	2.75
...
995	0.54	Ideal	...	5.34	3.26
996	0.72	Ideal	...	5.74	3.57
997	0.72	Good	...	5.89	3.48
998	0.74	Premium	...	5.77	3.58
999	1.12	Premium	...	6.61	4.03

1000 rows x 10 columns

```
diamonds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
#   Column      Non-Null Count  Dtype
---  -
0   carat       1000 non-null   float64
1   cut         1000 non-null   object
2   color       1000 non-null   object
3   clarity     1000 non-null   object
4   depth       1000 non-null   float64
5   table       1000 non-null   float64
6   price       1000 non-null   int64
7   x           1000 non-null   float64
8   y           1000 non-null   float64
9   z           1000 non-null   float64
dtypes: float64(6), int64(1), object(3)
memory usage: 78.2+ KB
```

讀取大型 CSV 檔案

```
diamonds.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1000 entries, 0 to 999  
Data columns (total 10 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   carat       1000 non-null   float64  
1   cut         1000 non-null   object  
2   color       1000 non-null   object  
3   clarity     1000 non-null   object  
4   depth       1000 non-null   float64  
5   table       1000 non-null   float64  
6   price       1000 non-null   int64  
7   x           1000 non-null   float64  
8   y           1000 non-null   float64  
9   z           1000 non-null   float64  
dtypes: float64(6), int64(1), object(3)  
memory usage: 78.2+ KB
```

```
diamonds2 = pd.read_csv('data/diamonds.csv', nrows=1000,  
                        dtype={'carat': np.float32, 'depth': np.float32,  
                               'table': np.float32, 'x': np.float32,  
                               'y': np.float32, 'z': np.float32,  
                               'price': np.int16})
```

```
diamonds2.info()
```

改變成低精度型別來降低記憶體使用量

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 1000 entries, 0 to 999  
Data columns (total 10 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   carat       1000 non-null   float32  
1   cut         1000 non-null   object  
2   color       1000 non-null   object  
3   clarity     1000 non-null   object  
4   depth       1000 non-null   float32  
5   table       1000 non-null   float32  
6   price       1000 non-null   int16  
7   x           1000 non-null   float32  
8   y           1000 non-null   float32  
9   z           1000 non-null   float32  
dtypes: float32(6), int16(1), object(3)  
memory usage: 49.0+ KB
```

讀取大型 CSV 檔案

diamonds.describe()						
	carat	depth	...	y	z	
count	1000.000000	1000.000000	...	1000.000000	1000.000000	
mean	0.689280	61.722800	...	5.599180	3.457530	
std	0.195291	1.758879	...	0.611974	0.389819	
min	0.200000	53.000000	...	3.750000	2.270000	
25%	0.700000	60.900000	...	5.630000	3.450000	
50%	0.710000	61.800000	...	5.760000	3.550000	
75%	0.790000	62.600000	...	5.910000	3.640000	
max	1.270000	69.500000	...	7.050000	4.330000	
8 rows × 7 columns						

diamonds2.describe()						
	carat	depth	...	y	z	
count	1000.000000	1000.000000	...	1000.000000	1000.000000	
mean	0.689281	61.722824	...	5.599180	3.457533	
std	0.195291	1.758878	...	0.611972	0.389819	
min	0.200000	53.000000	...	3.750000	2.270000	
25%	0.700000	60.900002	...	5.630000	3.450000	
50%	0.710000	61.799999	...	5.760000	3.550000	
75%	0.790000	62.599998	...	5.910000	3.640000	
max	1.270000	69.500000	...	7.050000	4.330000	
8 rows × 7 columns						

#改成低精度後，幾乎沒有差別，只有犧牲一點點精度，但省下了38%的記憶體。

讀取大型 CSV 檔案

針對 cut、color、clarity 等 object 型態欄位做查閱，發現值都很單純，改用 category 存，可節省記憶體。

```
diamonds2.cut.value_counts()
```

```
Ideal      333
Premium    290
Very Good  226
Good        89
Fair        62
Name: cut, dtype: int64
```

```
diamonds2.color.value_counts()
```

```
E      240
F      226
G      139
D      129
H      125
I       95
J       46
Name: color, dtype: int64
```

```
diamonds2.clarity.value_counts()
```

```
SI1      306
VS2      218
VS1      159
SI2      154
VVS2      62
VVS1      58
I1        29
IF         14
Name: clarity, dtype: int64
```

```
diamonds3 = pd.read_csv('data/diamonds.csv', nrows=1000,
                        dtype={'carat': np.float32, 'depth': np.float32,
                               'table': np.float32, 'x': np.float32,
                               'y': np.float32, 'z': np.float32,
                               'price': np.int16,
                               'cut': 'category', 'color': 'category',
                               'clarity': 'category'})
```

```
diamonds3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
#   Column      Non-Null Count  Dtype
---  -
0   carat      1000 non-null   float32
1   cut         1000 non-null   category
2   color       1000 non-null   category
3   clarity     1000 non-null   category
4   depth       1000 non-null   float32
5   table       1000 non-null   float32
6   price       1000 non-null   int16
7   x           1000 non-null   float32
8   y           1000 non-null   float32
9   z           1000 non-null   float32
dtypes: category(3), float32(6), int16(1)
memory usage: 29.4 KB
```

#再將 object 改成 category 後，僅佔原本 78kb 的 37%。

讀取大型 CSV 檔案

```
cols = ['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price']
diamonds4 = pd.read_csv('data/diamonds.csv', nrows=1000,
                        dtype={'carat': np.float32, 'depth': np.float32,
                               'table': np.float32, 'price': np.int16,
                               'cut': 'category', 'color': 'category',
                               'clarity': 'category'},
                        usecols=cols)
```

```
diamonds4.info()
```

#排除 x, y, z 欄位。

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  -
0   carat        1000 non-null   float32
1   cut          1000 non-null   category
2   color        1000 non-null   category
3   clarity      1000 non-null   category
4   depth        1000 non-null   float32
5   table        1000 non-null   float32
6   price        1000 non-null   int16
dtypes: category(3), float32(3), int16(1)
memory usage: 17.7 KB
```

讀取大型 CSV 檔案

```
cols = ['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'price']
diamonds_iter = pd.read_csv('data/diamonds.csv', nrows=1000,
                             dtype={'carat': np.float32, 'depth': np.float32,
                                     'table': np.float32, 'price': np.int16,
                                     'cut': 'category', 'color': 'category',
                                     'clarity': 'category'},
                             usecols=cols,
                             chunksize=200)

def process(df):
    return f'processed {df.size} items'

for chunk in diamonds_iter:
    print(process(chunk))
```

```
processed 1400 items
processed 1400 items
processed 1400 items
processed 1400 items
processed 1400 items
```

查詢值域及使用的記憶體大小

```
np.iinfo(np.int8)
```

```
iinfo(min=-128, max=127, dtype=int8)
```

```
diamonds4['price'].min()
```

```
326
```

```
diamonds4['price'].max()
```

```
2898
```

```
np.finfo(np.float16)
```

```
finfo(resolution=0.001, min=-6.55040e+04, max=6.55040e+04, dtype=float16)
```

```
diamonds.price.memory_usage()
```

```
8128
```

```
diamonds.price.memory_usage(index=False)
```

```
8000
```

```
diamonds.cut.memory_usage(deep=True)
```

```
63461
```

讀寫 EXCEL

```
!pip install xlwt
```

```
Collecting xlwt  
  Downloading xlwt-1.3.0-py2.py3-none-any.whl (99 kB)  
Installing collected packages: xlwt  
Successfully installed xlwt-1.3.0
```

```
WARNING: You are using pip version 21.1.1; however, version 22.2.2 is available.  
You should consider upgrading via the 'c:\users\test\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.
```

```
!pip install openpyxl
```

```
Collecting openpyxl  
  Downloading openpyxl-3.0.10-py2.py3-none-any.whl (242 kB)  
Collecting et_xmlfile  
  Using cached et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)  
Installing collected packages: et_xmlfile, openpyxl  
Successfully installed et_xmlfile-1.1.0 openpyxl-3.0.10
```

```
WARNING: You are using pip version 21.1.1; however, version 22.2.2 is available.  
You should consider upgrading via the 'c:\users\test\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.
```

```
!pip install xlrd
```

```
Collecting xlrd  
  Using cached xlrd-2.0.1-py2.py3-none-any.whl (96 kB)  
Installing collected packages: xlrd
```

```
WARNING: You are using pip version 21.1.1; however, version 22.2.2 is available.  
You should consider upgrading via the 'c:\users\test\appdata\local\programs\python\python38\python.exe -m pip install --upgrade pip' command.
```

```
Successfully installed xlrd-2.0.1
```

讀寫 EXCEL

```
beatles.to_excel('beat.xlsx')
```

```
beat2 = pd.read_excel('beat.xlsx')  
beat2
```

	Unnamed: 0	first	last	birth
0	0	Paul	McCartney	1942
1	1	John	Lennon	1940
2	2	Richard	Starkey	1940
3	3	George	Harrison	1943

```
beat2 = pd.read_excel('beat.xlsx', index_col=0)  
beat2
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

```
xl_writer = pd.ExcelWriter('beat1.xlsx')  
beatles.to_excel(xl_writer, sheet_name='All')  
beatles[beatles.birth < 1941].to_excel(xl_writer, sheet_name='1940')  
xl_writer.save()
```

	A	B	C	D	E
1		first	last	birth	
2	0	Paul	McCartney	1942	
3	1	John	Lennon	1940	
4	2	Richard	Starkey	1940	
5	3	George	Harrison	1943	
6					

Navigation: < > | All | 1940 | ⊕

	A	B	C	D	E
1		first	last	birth	
2	1	John	Lennon	1940	
3	2	Richard	Starkey	1940	
4					
5					
6					

Navigation: < > | All | 1940 | ⊕

讀取 zip 中的 csv

#zip 中，只有一個 csv 的情況

```
autos = pd.read_csv('data/vehicles.csv.zip')
autos
```

C:\Users\Admin\anaconda3\lib\site-packages\IPython\terminal\interactiveshell.py:333: DtypeWarning: Columns (10) have mixed types.Specify dtype option on import or pass low_memory=True.
has_raised = await self.run_ast_nodes(code_ast.body, in_namespace={})

	barrels08	barrelsA08	...	phevHwy	phevComb
0	15.695714	0.0	...	0	0
1	29.964545	0.0	...	0	0
2	12.207778	0.0	...	0	0
3	29.964545	0.0	...	0	0
4	17.347895	0.0	...	0	0
...
39096	14.982273	0.0	...	0	0
39097	14.330870	0.0	...	0	0
39098	15.695714	0.0	...	0	0
39099	15.695714	0.0	...	0	0
39100	18.311667	0.0	...	0	0

39101 rows x 83 columns

zip
中，
有多
個
csv
的
情況

```
import zipfile
```

```
with zipfile.ZipFile('data/kaggle-survey-2018.zip') as z:
    print('\n'.join(z.namelist()))
    kag = pd.read_csv(z.open('multipleChoiceResponses.csv'))
    kag_questions = kag.iloc[0]
    survey = kag.iloc[1:]
```

multipleChoiceResponses.csv
freeFormResponses.csv
SurveySchema.csv

```
survey.head(2).T
```

	1	2
Time from Start to Finish (seconds)	710	434
Q1	Female	Male
Q1_OTHER_TEXT	-1	-1
Q2	45-49	30-34
Q3	United S...	Indonesia
...
Q50_Part_5	NaN	NaN
Q50_Part_6	NaN	NaN
Q50_Part_7	NaN	NaN
Q50_Part_8	NaN	NaN
Q50_OTHER_TEXT	-1	-1

395 rows x 2 columns

存取資料庫

```
import sqlite3
con = sqlite3.connect('data/beat.db')
with con:
    cur = con.cursor()
    cur.execute("""DROP TABLE Band""")
    cur.execute("""CREATE TABLE Band(id INTEGER PRIMARY KEY,
                                     fname TEXT, lname TEXT, birthyear INT)""")
    cur.execute("""INSERT INTO Band VALUES(
                                     0, 'Paul', 'McCartney', 1942)""")
    cur.execute("""INSERT INTO Band VALUES(
                                     1, 'John', 'Lennon', 1940)""")
    _ = con.commit()
```

	fname	lname	birthyear
id			
0	Paul	McCartney	1942
1	John	Lennon	1940

```
sql = '''SELECT fname, birthyear from Band'''
fnames = pd.read_sql(sql, con)
fnames
```

	fname	birthyear
0	Paul	1942
1	John	1940

```
import sqlalchemy as sa
engine = sa.create_engine('sqlite:///data/beat.db', echo=True)
sa_connection = engine.connect()
```

```
beat = pd.read_sql('Band', sa_connection, index_col='id')
beat
```

```
2021-11-10 17:44:47,791 INFO sqlalchemy.engine.base.Engine SELECT CAST('test plain returns' AS VARCHAR(60)) AS anon_1
2021-11-10 17:44:47,793 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,795 INFO sqlalchemy.engine.base.Engine SELECT CAST('test unicode returns' AS VARCHAR(60)) AS anon_1
2021-11-10 17:44:47,797 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,799 INFO sqlalchemy.engine.base.Engine PRAGMA main.table_info("Band")
2021-11-10 17:44:47,801 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,804 INFO sqlalchemy.engine.base.Engine SELECT name FROM sqlite_master WHERE type='table' ORDER BY name
2021-11-10 17:44:47,805 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,890 INFO sqlalchemy.engine.base.Engine PRAGMA main.table_xinfo("Band")
2021-11-10 17:44:47,892 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,894 INFO sqlalchemy.engine.base.Engine SELECT sql FROM (SELECT * FROM sqlite_master UNION ALL SELECT * F
ROM sqlite_temp_master) WHERE name = 'Band' AND type = 'table'
2021-11-10 17:44:47,895 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,962 INFO sqlalchemy.engine.base.Engine PRAGMA main.foreign_key_list("Band")
2021-11-10 17:44:47,963 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,965 INFO sqlalchemy.engine.base.Engine PRAGMA temp.foreign_key_list("Band")
2021-11-10 17:44:47,965 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,967 INFO sqlalchemy.engine.base.Engine SELECT sql FROM (SELECT * FROM sqlite_master UNION ALL SELECT * F
ROM sqlite_temp_master) WHERE name = 'Band' AND type = 'table'
2021-11-10 17:44:47,969 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,973 INFO sqlalchemy.engine.base.Engine PRAGMA main.index_list("Band")
2021-11-10 17:44:47,975 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,977 INFO sqlalchemy.engine.base.Engine PRAGMA temp.index_list("Band")
2021-11-10 17:44:47,979 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,982 INFO sqlalchemy.engine.base.Engine PRAGMA main.index_list("Band")
2021-11-10 17:44:47,984 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,985 INFO sqlalchemy.engine.base.Engine PRAGMA temp.index_list("Band")
2021-11-10 17:44:47,987 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,989 INFO sqlalchemy.engine.base.Engine SELECT sql FROM (SELECT * FROM sqlite_master UNION ALL SELECT * F
ROM sqlite_temp_master) WHERE name = 'Band' AND type = 'table'
2021-11-10 17:44:47,990 INFO sqlalchemy.engine.base.Engine ()
2021-11-10 17:44:47,996 INFO sqlalchemy.engine.base.Engine SELECT "Band".id, "Band".fname, "Band".lname, "Band".birthyear
FROM "Band"
2021-11-10 17:44:47,997 INFO sqlalchemy.engine.base.Engine ()
```

存取 JSON

```
import pandas as pd
import numpy as np
pd.set_option('max_columns', 4, 'max_rows', 10, 'max_colwidth', 12)
```

```
fname = ['Paul', 'John', 'Richard', 'George']
lname = ['McCartney', 'Lennon', 'Starkey', 'Harrison']
birth = [1942, 1940, 1940, 1943]
```

```
people = {'first': fname, 'last': lname, 'birth': birth}
```

```
import json
encoded = json.dumps(people)
encoded
```

```
'{"first": ["Paul", "John", "Richard", "George"], "last": ["McCartney", "Lennon", "Starkey", "Harrison"], "birth": [1942, 1940, 1940, 1943]}'
```

```
json.loads(encoded)
```

```
{'first': ['Paul', 'John', 'Richard', 'George'],
 'last': ['McCartney', 'Lennon', 'Starkey', 'Harrison'],
 'birth': [1942, 1940, 1940, 1943]}
```

```
beatles = pd.read_json(encoded)
beatles
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

存取 JSON

```
records = beatles.to_json(orient='records')  
records
```

```
'[{"first": "Paul", "last": "McCartney", "birth": 1942}, {"first": "John", "last": "Lennon", "birth": 1940}, {"first": "Richard", "last": "Starkey", "birth": 1940}, {"first": "George", "last": "Harrison", "birth": 1943}]'
```

```
pd.read_json(records, orient='records')
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

存取 JSON

```
split = beatles.to_json(orient='split')
split
```

```
'{"columns":["first","last","birth"],"index":[0,1,2,3],"data":[["Paul","McCartney",1942],["John","Lennon",1940],["Richard","Starkey",1940],["George","Harrison",1943]]}'
```

```
pd.read_json(split, orient='split')
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

```
index = beatles.to_json(orient='index')
index
```

```
'{"0":{"first":"Paul","last":"McCartney","birth":1942},"1":{"first":"John","last":"Lennon","birth":1940},"2":{"first":"Richard","last":"Starkey","birth":1940},"3":{"first":"George","last":"Harrison","birth":1943}}'
```

```
pd.read_json(index, orient='index')
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

存取 JSON

```
values = beatles.to_json(orient='values')
values
```

```
'[["Paul", "McCartney", 1942], ["John", "Lennon", 1940], ["Richard", "Starkey", 1940], ["George", "Harrison", 1943]]'
```

```
pd.read_json(values, orient='values')
```

	0	1	2
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943

```
(pd.read_json(values, orient='values')
 .rename(columns=dict(enumerate(['first', 'last', 'birth']))))
)
```

	first	last	birth
0	Paul	McCartney	1942
1	John	Lennon	1940
2	Richard	Starkey	1940
3	George	Harrison	1943