

Anjuman-I-Islam's

AKBAR PEERBHOY COLLEGE OF COMMERCE AND ECONOMICS

M.S Road Do Taki, Mumbai- 400008

Data Science

Submitted by

Guided by – Prof. Hanna Saadya Shaikh

University of Mumbai

MSC IT (Information Technology) – Semester 1

Academic Year 2024-2025

INDEX

Practical No		Title	Date	Sign
Practical 1		Create a data model using Cassandra	08-10-23	
Practical 2		Conversion from different formats to HORUS	14-10-23	
	A	CSV to HORUS		
	B	XML to HORUS		
	C	JSON to HORUS		
	D	MySQL Database to HORUS		
	E	Picture to HORUS		
	F	Video to HORUS		
	G	Audio to HORUS		
Practical 3		Utilities and Auditing	21-10-23	
	A	Fixer Utilities		
	B	Data Binning or Bucketing		
	C	Aggregation of Data		
	D	Outlier Detection		
	E	Audit		
Practical 4		Retrieving Data	28-10-23	
	A	Data Processing		
	B	Retrieve different attributes of data		
	C	Data Pattern		
	D	Loading IP_DATA_ALL.csv		

	E	Building a diagram for scheduling of jobs		
	F	Connecting other Data Sources		
Practical 5		Assessing Data	04-11-23	
Practical 6		Processing Data	11-11-23	
Practical 7		Transforming Data	18-11-23	
Practical 8		Organizing Data	25-11-23	
	A	Horizontal Style		
	B	Vertical Style		
	C	Island Style		
	D	Secure Vault Style		
Practical 9		Reporting Data	02-12-23	
	A	Create a network routing diagram		
	B	Directed Acyclic Graph		
	C	Graphics		

Practical 1

Creating Data Model using Cassandra

Cassandra Data Model

Code:

```
Create keyspace keyspace1 with replication = {'class':'SimpleStrategy','replication_factor': 3};

Use keyspace1;

Create table dept ( dept_id int PRIMARY KEY, dept_name text, dept_loc text);

Create table emp ( emp_id int PRIMARY KEY, emp_name text, dept_id int, email text, phone text );

Insert into dept (dept_id, dept_name, dept_loc) values (1001, 'Accounts', 'Mumbai');

Insert into dept (dept_id, dept_name, dept_loc) values (1002, 'Marketing', 'Delhi');

Insert into dept (dept_id, dept_name, dept_loc) values (1003, 'HR', 'Chennai');

Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1001, 'ABCD',1001,
'abcd@company.com', '1122334455');

Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1002, 'DEFG',1001,
'defg@company.com', '2233445566');

Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1003, 'GHIJ',1002,
'ghij@company.com', '3344556677');

Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1004, 'JKLM',1002,
'jklm@company.com', '4455667788');

Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1005, 'MNOP',1003,
'mnop@company.com', '5566778899');

Insert into emp ( emp_id, emp_name, dept_id, email, phone ) values (1006, 'MNOP',1003,
'mnop@company.com', '5566778844');

select * from emp;

select * from dept;

update dept set dept_name='Human Resource' where dept_id=1003;

delete from emp where emp_id=1006;

select * from emp;
```

Output:

```

C:\Windows\system32\cmd.exe - cqlsh
cqlsh:keyspace1> Insert into emp ( emp_id, emp_name, dept_id, email, phone ) val
ues (1004, 'JKLM', 1002, 'jklm@company.com', '4455667788');
cqlsh:keyspace1> Insert into emp ( emp_id, emp_name, dept_id, email, phone ) val
ues (1005, 'MNOP', 1003, 'mnop@company.com', '5566778899');
cqlsh:keyspace1> Insert into emp ( emp_id, emp_name, dept_id, email, phone ) val
ues (1006, 'MNOP', 1003, 'mnop@company.com', '5566778844');
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email                | emp_name | phone
-----+-----+-----+-----+-----
  1006 |    1003 | mnop@company.com    | MNOP    | 5566778844
  1004 |    1002 | jklm@company.com    | JKLM    | 4455667788
  1005 |    1003 | mnop@company.com    | MNOP    | 5566778899
  1001 |    1001 | abcd@company.com    | ABCD    | 1122334455
  1003 |    1002 | ghij@company.com    | GHIJ    | 3344556677
  1002 |    1001 | defg@company.com    | DEFG    | 2233445566

(6 rows)
cqlsh:keyspace1> select * from dept;

 dept_id | dept_loc | dept_name
-----+-----+-----
   1001 |  Mumbai | Accounts
   1003 |  Chennai |      HR
   1002 |   Delhi | Marketing

(3 rows)
cqlsh:keyspace1> update dept set dept_name='Human Resource' where dept_id=1003;
cqlsh:keyspace1> select * from dept;

 dept_id | dept_loc | dept_name
-----+-----+-----
   1001 |  Mumbai | Accounts
   1003 |  Chennai | Human Resource
   1002 |   Delhi | Marketing

(3 rows)
cqlsh:keyspace1> delete from emp where emp_id=1006;
cqlsh:keyspace1> select * from emp;

 emp_id | dept_id | email                | emp_name | phone
-----+-----+-----+-----+-----
  1004 |    1002 | jklm@company.com    | JKLM    | 4455667788
  1005 |    1003 | mnop@company.com    | MNOP    | 5566778899
  1001 |    1001 | abcd@company.com    | ABCD    | 1122334455
  1003 |    1002 | ghij@company.com    | GHIJ    | 3344556677
  1002 |    1001 | defg@company.com    | DEFG    | 2233445566

(5 rows)

```

Practical 2

Conversion from different formats to HORUS formats

A) Csv to Horus

Code:

```
import pandas as pd
# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.csv'
InputData=pd.read_csv(sInputFileName,encoding="latin-1")
print('Input Data Values =====')
print(InputData)
print('=====')
# Processing Rules =====
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =====')
print(ProcessData)
print('=====')
# Output Agreement =====
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('CSV to HORUS - Done')
# Utility done
```

Output:

```
In [93]: runfile('H:/VKHCG/05-DS/9999-Data/CSV2HORUS.py', wdir='H:/VKHCG/05-DS/9999-Data')
```

```
Input Data Values =====
```

	Country	ISO-2-CODE	ISO-3-Code	ISO-M49
0	Afghanistan	AF	AFG	4
1	Aland Islands	AX	ALA	248
2	Albania	AL	ALB	8
3	Algeria	DZ	DZA	12
4	American Samoa	AS	ASM	16
5	Andorra	AD	AND	20
6	Angola	AO	AGO	24
7	Anguilla	AI	AIA	660
8	Antarctica	AQ	ATA	10
9	Antigua and Barbuda	AG	ATG	28
10	Argentina	AR	ARG	32
11	Armenia	AM	ARM	51
12	Aruba	AW	ABW	533
13	Australia	AU	AUS	36
14	Austria	AT	AUT	40
15	Azerbaijan	AZ	AZE	31
16	Bahamas	BS	BHS	44
17	Bahrain	BH	BHR	48
18	Bangladesh	BD	BGD	50
19	Barbados	BB	BRB	52
20	Belarus	BY	BLR	112
21	Belgium	BE	BEL	56
22	Belize	BZ	BLZ	84
23	Benin	BJ	BEN	204
24	Bermuda	BM	BMU	60
25	Bhutan	BT	BTN	64
26	Bolivia	BO	BOL	68
27	Bosnia and Herzegovina	BA	BIH	70
28	Botswana	BW	BWA	72
29	Bouvet Island	BV	BVT	74

217	Tajikistan	TJ	TJK	762
218	Tanzania, United Republic of	TZ	TZA	834
219	Thailand	TH	THA	764
220	Timor-Leste	TL	TLS	626
221	Togo	TG	TGO	768
222	Tokelau	TK	TKL	772
223	Tonga	TO	TON	776
224	Trinidad and Tobago	TT	TTO	780
225	Tunisia	TN	TUN	788
226	Turkey	TR	TUR	792
227	Turkmenistan	TM	TKM	795
228	Turks and Caicos Islands	TC	TCA	796
229	Tuvalu	TV	TUV	798
230	Uganda	UG	UGA	800
231	Ukraine	UA	UKR	804
232	United Arab Emirates	AE	ARE	784
233	United Kingdom	GB	GBR	826
234	United States of America	US	USA	840
235	US Minor Outlying Islands	UM	UMI	581
236	Uruguay	UY	URY	858
237	Uzbekistan	UZ	UZB	860

243	Western Sahara	EH	ESH	732
244	Yemen	YE	YEM	887
245	Zambia	ZM	ZMB	894
246	Zimbabwe	ZW	ZNE	716

```
[247 rows x 4 columns]
```

```
=====
```

```
Process Data Values =====
```

CountryNumber	CountryName
716	Zimbabwe
894	Zambia
887	Yemen
732	Western Sahara
876	Wallis and Futuna Islands
850	Virgin Islands, US
704	Viet Nam
862	Venezuelay(Bolivarian Republic)
548	Vanuatu
860	Uzbekistan
858	Uruguay
840	United States of America
826	United Kingdom
784	United Arab Emirates
804	Ukraine
800	Uganda
581	US Minor Outlying Islands
798	Tuvalu
796	Turks and Caicos Islands
795	Turkmenistan
792	Turkey
788	Tunisia
780	Trinidad and Tobago
776	Tonga
772	Tokelau
768	Togo
626	Timor-Leste
764	Thailand
834	Tanzania, United Republic of
762	Tajikistan
...	...
74	Bouvet Island
72	Botswana
70	Bosnia and Herzegovina
68	Bolivia
64	Bhutan

64	Bhutan
60	Bermuda
204	Benin
84	Belize
56	Belgium
112	Belarus
52	Barbados
50	Bangladesh
48	Bahrain

44	Bahamas
31	Azerbaijan
40	Austria
36	Australia
533	Aruba
51	Armenia
32	Argentina
28	Antigua and Barbuda
10	Antarctica
660	Anguilla
24	Angola
20	Andorra
16	American Samoa
12	Algeria
8	Albania
248	Aland Islands
4	Afghanistan

```
[247 rows x 1 columns]
```

```
=====
```

```
CSV to HORUS - Done
```

b) XML to Horus

Code:

```
import pandas as pd
import xml.etree.ElementTree as ET
def df2xml(data):
    header = data.columns
    root = ET.Element('root')
    for row in range(data.shape[0]):
        entry = ET.SubElement(root,'entry')
        for index in range(data.shape[1]):
            schild=str(header[index])
            child = ET.SubElement(entry, schild)
            if str(data[schild][row]) != 'nan':
                child.text = str(data[schild][row])
            else:
                child.text = 'n/a'
        entry.append(child)
    result = ET.tostring(root)
    return result
def xml2df(xml_data):
    root = ET.XML(xml_data)
    all_records = []
    for i, child in enumerate(root):
        record = { }
        for subchild in child:
            record[subchild.tag] = subchild.text
        all_records.append(record)
    return pd.DataFrame(all_records)
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.xml'
InputData = open(sInputFileName).read()
print('=====')
print('Input Data Values =====')
print('=====')
print(InputData)
print('=====')
#=====
# Processing Rules =====
#=====
ProcessDataXML=InputData
# XML to Data Frame
ProcessData=xml2df(ProcessDataXML)
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
```



```

ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('=====')
print('Process Data Values =====')
print('=====')
print(ProcessData)
print('=====')
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-XML-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====')
print('XML to HORUS - Done')
print('=====')
# Utility done =====

```

Output:

```
In [105]: runfile('H:/VKHCG/05-DS/9999-Data/XML2HORUS.py', wdir='H:/VKHCG/05-DS/9999-Data')
```

```
=====
Input Data Values =====>
<root><entry><Country>Afghanistan</Country><Country>Afghanistan</Country><ISO-2-CODE>AF/<br>ISO-2-CODE>ISO-2-CODE>AF/<ISO-2-CODE>ISO-2-CODE>ISO-2-CODE>AF&G/<ISO-3-CODE>ISO-3-CODE>AF&G/<ISO-3-<br>CODE>ISO-M49-44/<ISO-M49>ISO-M49-44/<ISO-M49-44/<ISO-M49></entry><entry><Country>Aland Islands</<br>Country><Country>Aland Islands</Country><ISO-2-CODE>AX/<ISO-2-CODE>ISO-2-CODE>AX/<ISO-2-<br>CODE>ISO-3-CODE>ALA/<ISO-3-CODE>ISO-3-CODE>ALA/<ISO-3-CODE>ISO-M49-248/<ISO-M49>ISO-M49-248/<ISO-M49></entry><entry><Country>Albania</Country><Country>Albania</Country><ISO-3-CODE>AL/<br>ISO-2-CODE>ISO-2-CODE>AL/<ISO-2-CODE>ISO-2-CODE>ISO-2-CODE>ISO-2-CODE>ALB/<ISO-3-CODE>ISO-3-CODE>ISO-3-CODE>ALB/<ISO-3-CODE>ISO-M49-87/<ISO-M49>ISO-M49-87/<ISO-M49></entry><entry><Country>Algeria</<br>Country><Country>Algeria</Country><ISO-2-CODE>DZ/<ISO-2-CODE>ISO-2-CODE>DZ/<ISO-2-CODE>ISO-2-CODE>ISO-2-CODE>DZA/<ISO-3-CODE>ISO-3-CODE>DZA/<ISO-3-CODE>ISO-M49-112/<ISO-M49>ISO-M49-112/<ISO-M49></entry><entry><Country>American Samoa</Country><Country>American Samoa</<br>Country><ISO-2-CODE>AS/<ISO-2-CODE>ISO-2-CODE>AS/<ISO-2-CODE>ISO-2-CODE>ISO-3-CODE>ASM/<ISO-3-<br>CODE>ISO-3-CODE>ASM/<ISO-3-CODE>ISO-3-CODE>ISO-M49-161/<ISO-M49>ISO-M49-161/<ISO-M49></entry><entry><Country>Andorra</Country><Country>Andorra</Country><ISO-2-CODE>AD/<ISO-2-CODE>ISO-2-CODE>AD/<ISO-2-CODE>ISO-2-CODE>ISO-2-CODE>AND/<ISO-3-CODE>ISO-3-CODE>AND/<ISO-3-CODE>ISO-M49-28/<ISO-M49>ISO-M49-28/<ISO-M49></entry><entry><Country>Angola</Country><Country>Angola</Country><ISO-2-CODE>AO/<ISO-2-CODE>ISO-2-CODE>AO/<ISO-2-CODE>ISO-2-CODE>ISO-2-CODE>AGO/<ISO-3-CODE>ISO-3-CODE>ISO-M49-24/<ISO-M49>ISO-M49-24/<ISO-M49></entry><entry><Country>Anguilla</Country><Country>Anguilla</Country><ISO-2-CODE>AI/<ISO-2-CODE>ISO-2-CODE>AI/<ISO-2-CODE>ISO-2-CODE>ISO-3-CODE>ATAI/<ISO-3-CODE>ISO-3-CODE>ATAI/<ISO-3-CODE>ISO-M49-660/<ISO-M49>ISO-M49-660/<ISO-M49></entry><entry><Country>Antarctica</Country><Country>Antarctica</Country><ISO-2-CODE>AQ/<ISO-2-CODE>ISO-2-CODE>AQ/<ISO-2-CODE>ISO-3-CODE>ISQ/<ISO-3-CODE>ISO-3-CODE>ISQ/<ISO-3-CODE>ISO-M49-10/<ISO-M49>ISO-M49-10/<ISO-M49></entry><entry><Country>Antigua and Barbuda</Country><Country>Antigua and Barbuda</Country><ISO-2-CODE>AG/<ISO-2-CODE>ISO-2-CODE>AG/<ISO-2-CODE>ISO-2-CODE>ATG/<ISO-3-CODE>ISO-3-CODE>ATG/<ISO-3-CODE>ISO-3-CODE>ISO-M49-28/<ISO-M49>ISO-M49-28/<ISO-M49></entry><entry><Country>Argentina</Country><Country>Argentina</Country><ISO-2-CODE>AR/<ISO-2-CODE>ISO-2-CODE>AR/<ISO-2-CODE>ISO-3-CODE>ISO-3-CODE>ARG/<ISO-3-CODE>ISO-3-CODE>ARG/<ISO-3-CODE>ISO-M49-32/<ISO-M49>ISO-M49-32/<ISO-M49></entry><entry><Country>Armenia</Country><Country>Armenia</Country><ISO-2-CODE>AM/<ISO-2-CODE>ISO-2-CODE>AM/<ISO-2-CODE>ISO-3-CODE>ARMK/<ISO-3-CODE>ISO-3-CODE>ARMK/<ISO-3-CODE>ISO-M49-51/<ISO-M49>ISO-M49-51/<ISO-M49></entry><entry><Country>Aruba</Country><Country>Aruba</Country><ISO-2-CODE>AW/<ISO-2-CODE>ISO-2-CODE>AW/<ISO-2-CODE>ISO-3-CODE>ABWK/<ISO-3-CODE>ISO-3-CODE>ABWK/<ISO-3-CODE>ISO-M49-533/<ISO-M49>ISO-M49-533/<ISO-M49></entry><entry><Country>Australia</Country>
```

[illegible]

```
Code<ISO-M49:52/ISO-M49:52/ISO-M49:52/ISO-M49:52/entry><entry>Country<Belarus/ISO-
Country<Belarus/ISO-Codes<ISO-2-Codes8Y/ISO-Codes<ISO-2-Codes8Y/ISO-
Country<Code8LR/ISO-Codes<ISO-3-Codes<ISO-3-Codes8LR/ISO-Codes<ISO-M49:112/ISO-M49:
M49:112/ISO-M49:112/entry><entry>Country<Belgium/Country<Country<Belgium/Country<ISO-2-
Code8E/ISO-Codes<ISO-2-Codes8E/ISO-Codes<ISO-3-Codes8E/ISO-Codes<ISO-3-Codes8E/ISO-
ISO-3-Codes<ISO-M49:56/ISO-M49:ISO-M49:56/ISO-M49:56/entry><entry>Country<Belize/
Country<Country<Belize/Country<ISO-Codes8Z/ISO-Codes<ISO-2-Codes8Z/ISO-Codes<ISO-2-
Codes8Z/ISO-2-
Codes<ISO-3-Codes8Z/ISO-Codes<ISO-3-Codes8Z/ISO-Codes<ISO-M49:84/ISO-M49:ISO-
M49:84/ISO-M49:84/entry><entry>Country<Benin/Country<Country<Benin/Country<ISO-2-
Code8S/ISO-Codes<ISO-2-Codes8S/ISO-Codes<ISO-3-Codes8S/ISO-Codes<ISO-3-Codes8S/ISO-
ISO-3-Codes<ISO-M49:56/ISO-M49:ISO-M49:56/ISO-M49:56/entry><entry>Country<Bermuda/
Country<Country<Bermuda/Country<ISO-2-Codes8W/ISO-Codes<ISO-2-Codes8W/ISO-Codes<
ISO-3-Codes8W/ISO-Codes<ISO-3-Codes8W/ISO-Codes<ISO-M49:68/ISO-M49:ISO-M49:68/ISO-M49:68/ISO-
```

Country<ISO-2-Code>CV</ISO-2-Code><ISO-2-Code>CV</ISO-2-Code><ISO-3-Code>CPV</ISO-3-Code><ISO-3-Code>CPV</ISO-3-Code><ISO-M49>132</ISO-M49><ISO-M49>132</ISO-M49></entry><entry><Country>Cayman Islands</Country><Country>Cayman Islands</Country><ISO-2-Code>KY</ISO-2-Code><ISO-2-Code>KY</ISO-2-Code><ISO-3-Code>CPK</ISO-3-Code><ISO-3-Code>CPK</ISO-3-Code><ISO-M49>136</ISO-M49><ISO-M49>136</ISO-M49></entry><entry><Country>Central

2

African Republic/<Country>Central African Republic/<Country>ISO-2-CODE/CF/Iso-2-CODE/ISO-2-CODE/CF/Iso-2-CODE/ISO-3-CODE/CF/Iso-3-CODE/ISO-3-CODE/CF/Iso-3-CODE/ISO-499/148/Iso-M49/Iso-499/148/Iso-M49/<entry>entry</Country>Chad/
Country>Country>Chad/Country>ISO-2-CODE/TD/Iso-2-CODE/ISO-2-CODE/TD/Iso-2-CODE/ISO-3-CODE/TD/Iso-3-CODE/ISO-3-CODE/TD/Iso-3-CODE/ISO-M49/148/Iso-M49/148/Iso-M49/<entry>entry</Country>Chile/Country>Country>Chile/Country>ISO-2-CODE/CL/Iso-2-CODE/ISO-2-CODE/CL/Iso-2-CODE/ISO-3-CODE/CL/Iso-3-CODE/ISO-3-CODE/CHL/Iso-3-CODE/ISO-3-CODE/CHL/Iso-3-CODE/ISO-M49/152/Iso-M49/152/Iso-M49/<entry>entry</Country>China/
Country>Country>China/Country>ISO-2-CODE/CN/Iso-2-CODE/ISO-2-CODE/CN/Iso-2-CODE/ISO-3-CODE/CN/Iso-3-CODE/ISO-3-CODE/CN/Iso-3-CODE/ISO-M49/156/Iso-M49/156/Iso-M49/<entry>entry</Country>Hong Kong, SAR China/Country>Country>Hong Kong, SAR China/Country>ISO-2-CODE/HK/Iso-2-CODE/ISO-2-CODE/HK/Iso-2-CODE/ISO-3-CODE/HK/Iso-3-CODE/ISO-3-CODE/HKG/Iso-3-CODE/ISO-3-CODE/HKG/Iso-3-CODE/ISO-M49/344/Iso-M49/344/Iso-M49/<entry>entry</Country>Macao, SAR China/Country>Country>Macao, SAR China/Country>ISO-2-CODE/MO/Iso-2-CODE/ISO-2-CODE/MO/Iso-2-CODE/ISO-3-CODE/MAC/Iso-3-CODE/ISO-3-CODE/MAC/Iso-3-CODE/ISO-M49/446/Iso-M49/446/Iso-M49/<entry>entry</Country>Christmas Island/Country>Country>Christmas Island/Country>ISO-2-CODE/CK/Iso-2-CODE/CK/Iso-2-CODE/CK/Iso-2-CODE/ISO-3-CODE/CKR/Iso-3-CODE/ISO-3-CODE/CKR/Iso-3-CODE/ISO-M49/162/

[illegible]

c) JSON to HORUS Format

Code:

```
# Utility Start JSON to HORUS =====
# Standard Tools
#=====
import pandas as pd
# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/Country_Code.json'
InputData=pd.read_json(sInputFileName, orient='index', encoding="latin-1")
print('Input Data Values =====')
print(InputData)
print('=====')
# Processing Rules =====
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =====')
print(ProcessData)
print('=====')
# Output Agreement =====
OutputData=ProcessData
sOutputFileName='c:/VKHCG/05-DS/9999-Data/HORUS-JSON-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('JSON to HORUS - Done')
# Utility done =====
```

Output:

```

In [117]: runfile('H:/VKHCG/05-DS/9999-Data/JSON2HORUS.py', wdir='H:/VKHCG/05-DS/9999-Data')

Input Data Values =====
0          Country ISO-2-CODE ISO-3-Code ISO-M49
1      Afghanistan AF AFG 4
2      Aland Islands AX ALA 248
3      Albania AL ALB 8
4      Algeria DZ DZA 12
5      American Samoa AS ASM 16
6      Andorra AD AND 20
7      Angola AO AGO 24
8      Anguilla AI AIA 660
9      Antarctica AQ ATA 10
10     Antigua and Barbuda AG ATG 28
11     Argentina AR ARG 32
12     Armenia AM ARM 51
13     Aruba AW ABW 533
14     Australia AU AUS 36
15     Austria AT AUT 40
16     Azerbaijan AZ AZE 31
17     Bahamas BS BHS 44
18     Bahrain BH BHR 48
19     Bangladesh BD BGD 50
20     Barbados BB BRB 52
21     Belarus BY BLR 112
22     Belgium BE BEL 56
23     Belize BZ BLZ 84
24     Benin BJ BEN 204
25     Bermuda BM BMU 60
26     Bhutan BT BTN 64
27     Bolivia BO BOL 68
28     Bosnia and Herzegovina BA BIH 70
29     Botswana BW BWA 72
30     Bouvet Island BV BVT 74
31     ... ..
32     Tajikistan TJ TJK 762
33     Tanzania, United Republic of TZ TZA 834
34     Thailand TH THA 764
35     Timor-Leste TL TLS 626
36     Togo TG TGO 768
37     Tokelau TK TKL 772

223     Tonga TO TON 776
224     Trinidad and Tobago TT TTO 780
225     Tunisia TN TUN 788
226     Turkey TR TUR 792
227     Turkmenistan TM TKM 795
228     Turks and Caicos Islands TC TCA 796
229     Tuvalu TV TUV 798
230     Uganda UG UGA 800
231     Ukraine UA UKR 804
232     United Arab Emirates AE ARE 784
233     United Kingdom GB GBR 826
234     United States of America US USA 840
235     US Minor Outlying Islands UM UMI 581
236     Uruguay UY URY 858
237     Uzbekistan UZ UZB 860

238     Vanuatu VU VUT 548
239     Venezuelay(Bolivarian Republic) VE VEN 862
240     Viet Nam VN VNM 704
241     Virgin Islands, US VI VIR 850
242     Wallis and Futuna Islands WF WLF 876
243     Western Sahara EH ESH 732
244     Yemen YE YEM 887
245     Zambia ZM ZMB 894
246     Zimbabwe ZW ZWE 716

[247 rows x 4 columns]
=====

Process Data Values =====
CountryNumber CountryName
716 Zimbabwe
894 Zambia
887 Yemen 64 Bhutan
732 Western Sahara 60 Bermuda
876 Wallis and Futuna Islands 204 Benin
850 Virgin Islands, US 84 Belize
704 Viet Nam 56 Belgium
862 Venezuelay(Bolivarian Republic) 112 Belarus
548 Vanuatu 52 Barbados
860 Uzbekistan 48 Bangladesh
858 Uruguay Bahrain
840 United States of America
826 United Kingdom
784 United Arab Emirates
804 Ukraine
800 Uganda
581 US Minor Outlying Islands
798 Tuvalu
796 Turks and Caicos Islands
795 Turkmenistan
792 Turkey
788 Tunisia 44 Bahamas
780 Trinidad and Tobago 31 Azerbaijan
776 Tonga 40 Austria
772 Tokelau 36 Australia
768 Togo 533 Aruba
626 Timor-Leste 51 Armenia
764 Thailand 50 Argentina
834 Tanzania, United Republic of 32 Antigua and Barbuda
762 Tajikistan 28 Antarctica
... .. 24 Anguilla
74 Bouvet Island 20 Angola
72 Botswana 16 Andorra
70 Bosnia and Herzegovina 12 American Samoa
68 Bolivia 8 Algeria
64 Bhutan 248 Albania
60 Bermuda 4 Aland Islands
204 Benin
84 Belize

[247 rows x 1 columns]
=====
JSON to HORUS - Done

```

d) MySql Database to HORUS Format

Code :

```
import pandas as pd
import sqlite3 as sq
# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/utility.db'
sInputTable='Country_Code'
conn = sq.connect(sInputFileName)
sSQL='select * FROM ' + sInputTable + ';'
InputData=pd.read_sql_query(sSQL, conn)
print('Input Data Values =====')
print(InputData)
print('=====')
# Processing Rules =====
ProcessData=InputData
# Remove columns ISO-2-Code and ISO-3-CODE
ProcessData.drop('ISO-2-CODE', axis=1,inplace=True)
ProcessData.drop('ISO-3-Code', axis=1,inplace=True)
# Rename Country and ISO-M49
ProcessData.rename(columns={'Country': 'CountryName'}, inplace=True)
ProcessData.rename(columns={'ISO-M49': 'CountryNumber'}, inplace=True)
# Set new Index
ProcessData.set_index('CountryNumber', inplace=True)
# Sort data by CurrencyNumber
ProcessData.sort_values('CountryName', axis=0, ascending=False, inplace=True)
print('Process Data Values =====')
print(ProcessData)
print('=====')
# Output Agreement =====
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-CSV-Country.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('Database to HORUS - Done')
# Utility done =====
```

Output:

```
In [6]: runfile('C:/VKHCG/05-DS/9999-Data/DATABASE2HORUS.py', wdir='C:/VKHCG/05-DS/9999-Data')
```

```
Input Data Values =====
index      Country ISO-2-CODE ISO-3-Code ISO-M49
0          0          Afghanistan      AF      AFG      4
1          1          Aland Islands     AX      ALA      248
2          2          Albania           AL      ALB      8
3          3          Algeria           DZ      DZA      12
4          4          American Samoa    AS      ASM      16
5          5          Andorra           AD      AND      20
6          6          Angola           AO      AGO      24
7          7          Anguilla          AI      AIA      660
8          8          Antarctica        AQ      ATA      10
9          9          Antigua and Barbuda AG      ATG      28
10         10          Argentina         AR      ARG      32
11         11          Armenia           AM      ARM      51
12         12          Aruba             AW      ABW      533
13         13          Australia         AU      AUS      36
14         14          Austria           AT      AUT      40
15         15          Azerbaijan        AZ      AZE      31
16         16          Bahamas          BS      BHS      44
17         17          Bahrain           BH      BHR      48
18         18          Bangladesh        BD      BGD      50
19         19          Barbados          BB      BRB      52
20         20          Belarus           BY      BLR      112
21         21          Belgium           BE      BEL      56
22         22          Belize           BZ      BLZ      84
23         23          Benin             BJ      BEN      204
24         24          Bermuda           BM      BMU      60
25         25          Bhutan            BT      BTN      64
26         26          Bolivia           BO      BOL      68
27         27          Bosnia and Herzegovina BA      BIH      70
28         28          Botswana          BW      BWA      72

29         29          Bouvet Island     BV      BVT      74
...         ...          ...            ...      ...      ...
217        217          Tajikistan        TJ      TJK      762
218        218          Tanzania, United Republic of TZ      TZA      834
219        219          Thailand         TH      THA      764
220        220          Timor-Leste       TL      TLS      626
221        221          Togo              TG      TGO      768
222        222          Tokelau           TK      TKL      772
223        223          Tonga             TO      TON      776
224        224          Trinidad and Tobago TT      TTO      780
225        225          Tunisia           TN      TUN      788
226        226          Turkey            TR      TUR      792
227        227          Turkmenistan       TM      TKM      795
228        228          Turks and Caicos Islands TC      TCA      796
229        229          Tuvalu            TV      TUV      798
230        230          Uganda           UG      UGA      800
231        231          Ukraine           UA      UKR      804
232        232          United Arab Emirates AE      ARE      784
233        233          United Kingdom    GB      GBR      826
234        234          United States of America US      USA      840
235        235          US Minor Outlying Islands UM      UMI      581
236        236          Uruguay           UY      URY      858
237        237          Uzbekistan        UZ      UZB      860
238        238          Vanuatu           VU      VUT      548
239        239          Venezuela(Bolivarian Republic) VE      VEN      862
240        240          Viet Nam         VN      VNM      704
241        241          Virgin Islands, US VI      VIR      850
242        242          Wallis and Futuna Islands WF      WLF      876
243        243          Western Sahara    EH      ESH      732
244        244          Yemen            YE      YEM      887
245        245          Zambia            ZM      ZMB      894
246        246          Zimbabwe          ZW      ZWE      716
```

```
[247 rows x 5 columns]
```

```

Process Data Values =====
index          CountryName
CountryNumber
716            246            Zimbabwe
894            245            Zambia
887            244            Yemen
732            243            Western Sahara
876            242            Wallis and Futuna Islands
850            241            Virgin Islands, US
704            240            Viet Nam
862            239            Venezuelay(Bolivarian Republic)
548            238            Vanuatu
860            237            Uzbekistan
858            236            Uruguay
840            234            United States of America
826            233            United Kingdom
784            232            United Arab Emirates
804            231            Ukraine
800            230            Uganda
581            235            US Minor Outlying Islands
798            229            Tuvalu
796            228            Turks and Caicos Islands
795            227            Turkmenistan
792            226            Turkey
788            225            Tunisia
780            224            Trinidad and Tobago
776            223            Tonga
772            222            Tokelau
768            221            Togo
626            220            Timor-Leste
764            219            Thailand
834            218            Tanzania, United Republic of
762            217            Tajikistan
...            ...            ...
74            29            Bouvet Island
72            28            Botswana
70            27            Bosnia and Herzegovina

70            27            Bosnia and Herzegovina
68            26            Bolivia
64            25            Bhutan
60            24            Bermuda
204           23            Benin
84            22            Belize
56            21            Belgium
112           20            Belarus
52            19            Barbados
50            18            Bangladesh
48            17            Bahrain
44            16            Bahamas
31            15            Azerbaijan
40            14            Austria
36            13            Australia
533           12            Aruba
51            11            Armenia
32            10            Argentina
28            9            Antigua and Barbuda
10            8            Antarctica
660           7            Anguilla
24            6            Angola
20            5            Andorra
16            4            American Samoa
12            3            Algeria
8             2            Albania
248           1            Aland Islands
4             0            Afghanistan

[247 rows x 2 columns]
=====
Database to HORUS - Done

```

e) Picture (JPEG) to HORUS Format

Code:

```
# Utility Start Picture to HORUS =====
# Standard Tools
#=====
from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Input Agreement =====
sInputFileName='C:/VKHCG/05-DS/9999-Data/Angus.jpg'
InputData = imread(sInputFileName, flatten=False, mode='RGBA')
print('Input Data Values =====')
print('X: ',InputData.shape[0])
print('Y: ',InputData.shape[1])
print('RGBA: ', InputData.shape[2])
print('=====')
# Processing Rules =====
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha']
ProcessData.columns=sColumns
ProcessData.index.names =['ID']
print('Rows: ',ProcessData.shape[0])
print('Columns :',ProcessData.shape[1])
print('=====')
print('Process Data Values =====')
print('=====')
plt.imshow(InputData)
plt.show()
print('=====')
# Output Agreement =====
OutputData=ProcessData
print('Storing File')
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Picture.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====')
print('Picture to HORUS - Done')
print('=====')
```


f) i. Video to HORUS Format

Code:

Movies to Frames

```
import os
import shutil
import cv2
#=====
sInputFileName='C:/VKHCG/05-DS/9999-Data/dog.mp4'
sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp'
if os.path.exists(sDataBaseDir):
    shutil.rmtree(sDataBaseDir)
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
print('=====')
print('Start Movie to Frames')
print('=====')
vidcap = cv2.VideoCapture(sInputFileName)
success,image = vidcap.read()
count = 0
while success:
    success,image = vidcap.read()
    sFrame=sDataBaseDir + str('/dog-frame-' + str(format(count, '04d'))+ '.jpg')
    print('Extracted: ', sFrame)
    cv2.imwrite(sFrame, image)
    if os.path.getsize(sFrame) == 0:
        count += -1
    os.remove(sFrame)
    print('Removed: ', sFrame)
    if cv2.waitKey(10) == 27: # exit if Escape is hit
        break
    count += 1
print('=====')
print('Generated : ', count, ' Frames')
print('=====')
print('Movie to Frames HORUS - Done')
print('=====')
# Utility done =====
```

Output:

```
In [72]: runfile('C:/VKHCG/05-DS/9999-Data/MOVIE2HORUSFrame.py', wdir='C:/VKHCG/05-DS/9999-Data')
```

```
=====
Start Movie to Frames
=====
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0000.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0001.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0002.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0003.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0004.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0005.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0006.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0007.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0008.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0009.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0010.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0011.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0012.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0013.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0014.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0015.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0016.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0017.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0018.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0019.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0020.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0021.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0022.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0023.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0024.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0025.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0026.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0027.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0028.jpg

Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0079.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0080.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0081.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0082.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0083.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0084.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0085.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0086.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0087.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0088.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0089.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0090.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0091.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0092.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0093.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0094.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0095.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0096.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0097.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0098.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0099.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0100.jpg
Extracted: C:/VKHCG/05-DS/9999-Data/temp/dog-frame-0101.jpg
=====
Generated : 101 Frames
=====
Movie to Frames HORUS - Done
=====
```

ii. Frames to Horus

```

from scipy.misc import imread
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
# Input Agreement =====
sDataBaseDir='C:/VKHCG/05-DS/9999-Data/temp'
f=0
for file in os.listdir(sDataBaseDir):
    if file.endswith(".jpg"):
        f += 1
        sInputFileName=os.path.join(sDataBaseDir, file)
        print('Process : ', sInputFileName)
        InputData = imread(sInputFileName, flatten=False, mode='RGBA')
        print('Input Data Values =====')
        print('X: ',InputData.shape[0])
        print('Y: ',InputData.shape[1])
        print('RGBA: ', InputData.shape[2])
        print('=====')
# Processing Rules =====
ProcessRawData=InputData.flatten()
y=InputData.shape[2] + 2
x=int(ProcessRawData.shape[0]/y)
ProcessFrameData=pd.DataFrame(np.reshape(ProcessRawData, (x, y)))
ProcessFrameData['Frame']=file
print('=====')
print('Process Data Values =====')
print('=====')
plt.imshow(InputData)
plt.show()
if f == 1:
    ProcessData=ProcessFrameData
else:
    ProcessData=ProcessData.append(ProcessFrameData)
if f > 0:
    sColumns= ['XAxis','YAxis','Red', 'Green', 'Blue','Alpha','FrameName']
    ProcessData.columns=sColumns
    print('=====')
    ProcessFrameData.index.names =['ID']
    print('Rows: ',ProcessData.shape[0])
    print('Columns :',ProcessData.shape[1])
    print('=====')
# Output Agreement =====

```

g) Audio to HORUS Format

Code:

```
# Utility Start Audio to HORUS =====
# Standard Tools
#=====
from scipy.io import wavfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
#=====
def show_info(aname, a,r):
    print ('-----')
    print ("Audio:", aname)
    print ('-----')
    print ("Rate:", r)
    print ('-----')
    print ("shape:", a.shape)
    print ("dtype:", a.dtype)
    print ("min, max:", a.min(), a.max())
    print ('-----')
    plot_info(aname, a,r)
#=====
def plot_info(aname, a,r):
    sTitle= 'Signal Wave - '+ aname + ' at ' + str(r) + 'hz'
    plt.title(sTitle)
    sLegend=[]
    for c in range(a.shape[1]):
        sLabel = 'Ch' + str(c+1)
        sLegend=sLegend+[str(c+1)]
    plt.plot(a[:,c], label=sLabel)
    plt.legend(sLegend)
    plt.show()
#=====
sInputFileName='C:/VKHCG/05-DS/9999-Data/2ch-sound.wav'
print('=====')
print('Processing : ', sInputFileName)
print('=====')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("2 channel", InputData,InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-2ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
```

```

#=====
sInputFileName='C:/VKHCG/05-DS/9999-Data/4ch-sound.wav'
print('=====')
print('Processing : ', sInputFileName)
print('=====')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("4 channel", InputData, InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-4ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#=====
sInputFileName='C:/VKHCG/05-DS/9999-Data/6ch-sound.wav'
print('=====')
print('Processing : ', sInputFileName)
print('=====')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("6 channel", InputData, InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-6ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
#=====
sInputFileName='C:/VKHCG/05-DS/9999-Data/8ch-sound.wav'
print('=====')
print('Processing : ', sInputFileName)
print('=====')
InputRate, InputData = wavfile.read(sInputFileName)
show_info("8 channel", InputData, InputRate)
ProcessData=pd.DataFrame(InputData)
sColumns= ['Ch1','Ch2','Ch3', 'Ch4', 'Ch5','Ch6','Ch7','Ch8']
ProcessData.columns=sColumns
OutputData=ProcessData
sOutputFileName='C:/VKHCG/05-DS/9999-Data/HORUS-Audio-8ch.csv'
OutputData.to_csv(sOutputFileName, index = False)
print('=====')
print('Audio to HORUS - Done')

```

Practical 3

Utilities and Auditing

A. Fixers Utilities:

Fixers enable your solution to take your existing data and fix a specific quality issue.

#----- Program to Demonstrate Fixers utilities -----

```
import string
```

```
import datetime as dt
```

1 Removing leading or lagging spaces from a data entry

```
print('#1 Removing leading or lagging spaces from a data entry');
```

```
baddata = " Data Science with too many spaces is bad!!! "
```

```
print('>',baddata,<')
```

```
cleandata=baddata.strip()
```

```
print('>',cleandata,<')
```

2 Removing nonprintable characters from a data entry

```
print('#2 Removing nonprintable characters from a data entry')
```

```
printable = set(string.printable)
```

```
baddata = "Data\x00Science with\x02 funny characters is \x10bad!!!"
```

```
cleandata="".join(filter(lambda x: x in string.printable,baddata))
```

```
print('Bad Data : ',baddata);
```

```
print('Clean Data : ',cleandata)
```

3 Reformatting data entry to match specific formatting criteria.

Convert YYYY/MM/DD to DD Month YYYY

```
print('# 3 Reformatting data entry to match specific formatting criteria.')
```

```
baddate = dt.date(2019, 10, 31)
```

```
baddata=format(baddate,'%Y-%m-%d')
```

```
gooddate = dt.datetime.strptime(baddata,'%Y-%m-%d')
```

```
gooddata=format(gooddate,'%d %B %Y')
```

```
print('Bad Data : ',baddata)
```

```
print('Good Data : ',gooddata)
```

Output:

```
In [10]: runfile('C:/Users/MSGIT/.spyder-py3/temp.py',  
wdir='C:/Users/MSGIT/.spyder-py3')  
  
#1 Removing leading or lagging spaces from a data entry  
> Data Science with too many spaces is bad!!! <  
> Data Science with too many spaces is bad!!! <  
  
#2 Removing nonprintable characters from a data entry  
Bad Data : Data Science with funny characters is bad!!!  
Clean Data : DataScience with funny characters is bad!!!  
  
# 3 Reformatting data entry to match specific formatting  
criteria.  
Bad Data : 2019-10-31  
Good Data : 31 October 2019
```

B. Averaging of Data

Code:

```
import pandas as pd
#####
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ')
print('#####')
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
AllData=IP_DATA_ALL[['Country', 'Place_Name','Latitude']]
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
print(MeanData)
```

Output:

```
3561      DE      Munich      48.1480
```

```
[3562 rows x 3 columns]
```

```
Country  Place_Name
```

```
DE      Munich      48.143223
```

```
GB      London      51.509406
```

```
US      New York     40.747044
```

```
Name: Latitude, dtype: float64
```


D. Outlier Detection

Code:

```
#####
# -*- coding: utf-8 -*-
#####
import pandas as pd
#####
InputFileName='IP_DATA_CORE.csv'
OutputFileName='Retrieve_Router_Location.csv'
Base='C:/VKHCG'
print('#####')
print('Working Base :',Base)
print('#####')
#####
sFileName=Base + '/01-Vermeulen/00-RawData/' + InputFileName
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False,
usecols=['Country','Place Name','Latitude','Longitude'], encoding="latin-1")
IP_DATA_ALL.rename(columns={'Place Name': 'Place_Name'}, inplace=True)
LondonData=IP_DATA_ALL.loc[IP_DATA_ALL['Place_Name']=='London']
AllData=LondonData[['Country', 'Place_Name','Latitude']]
print('All Data')
print(AllData)
MeanData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].mean()
StdData=AllData.groupby(['Country', 'Place_Name'])['Latitude'].std()
print('Outliers')
UpperBound=float(MeanData+StdData)
print('Higher than ', UpperBound)
OutliersHigher=AllData[AllData.Latitude>UpperBound]
print(OutliersHigher)
LowerBound=float(MeanData-StdData)
print('Lower than ', LowerBound)
OutliersLower=AllData[AllData.Latitude<LowerBound]
print(OutliersLower)
print('Not Outliers')
OutliersNot=AllData[(AllData.Latitude>=LowerBound) & (AllData.Latitude<=UpperBound)]
print(OutliersNot)
```

Output:

```
In [35]: runfile('C:/Users/MSCIT/.spyder-py3/temp.py',
wdir='C:/Users/MSCIT/.spyder-py3')
```

#####

Working Base : C:/VKHCG

#####

Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_CORE.csv

All Data

	Country	Place_Name	Latitude
1910	GB	London	51.5130
1911	GB	London	51.5508
1912	GB	London	51.5649
1913	GB	London	51.5895
1914	GB	London	51.5232
1916	GB	London	51.5491
1919	GB	London	51.5161
1920	GB	London	51.5198
1921	GB	London	51.5198
1923	GB	London	51.5237
1924	GB	London	51.5237
1925	GB	London	51.5237
1926	GB	London	51.5237
1927	GB	London	51.5232
3436	GB	London	51.5163
3438	GB	London	51.5136

Lower than 51.50617687562166

	Country	Place_Name	Latitude
1915	GB	London	51.4739

Not Outliers

	Country	Place_Name	Latitude
1917	GB	London	51.5085
1918	GB	London	51.5085
1922	GB	London	51.5085
1928	GB	London	51.5085
1929	GB	London	51.5085
1957	GB	London	51.5115
1958	GB	London	51.5092
1959	GB	London	51.5092
1960	GB	London	51.5092
1961	GB	London	51.5092
1962	GB	London	51.5092

E. Logging

Code:

```
import sys
import os
import logging
import uuid
import shutil
import time

if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'

sCompanies=['01-Vermeulen','02-Krennwallner','03-Hillman','04-Clark']
sLayers=['01-Retrieve','02-Assess','03-Process','04-Transform','05-Organise','06-Report']
sLevels=['debug','info','warning','error']

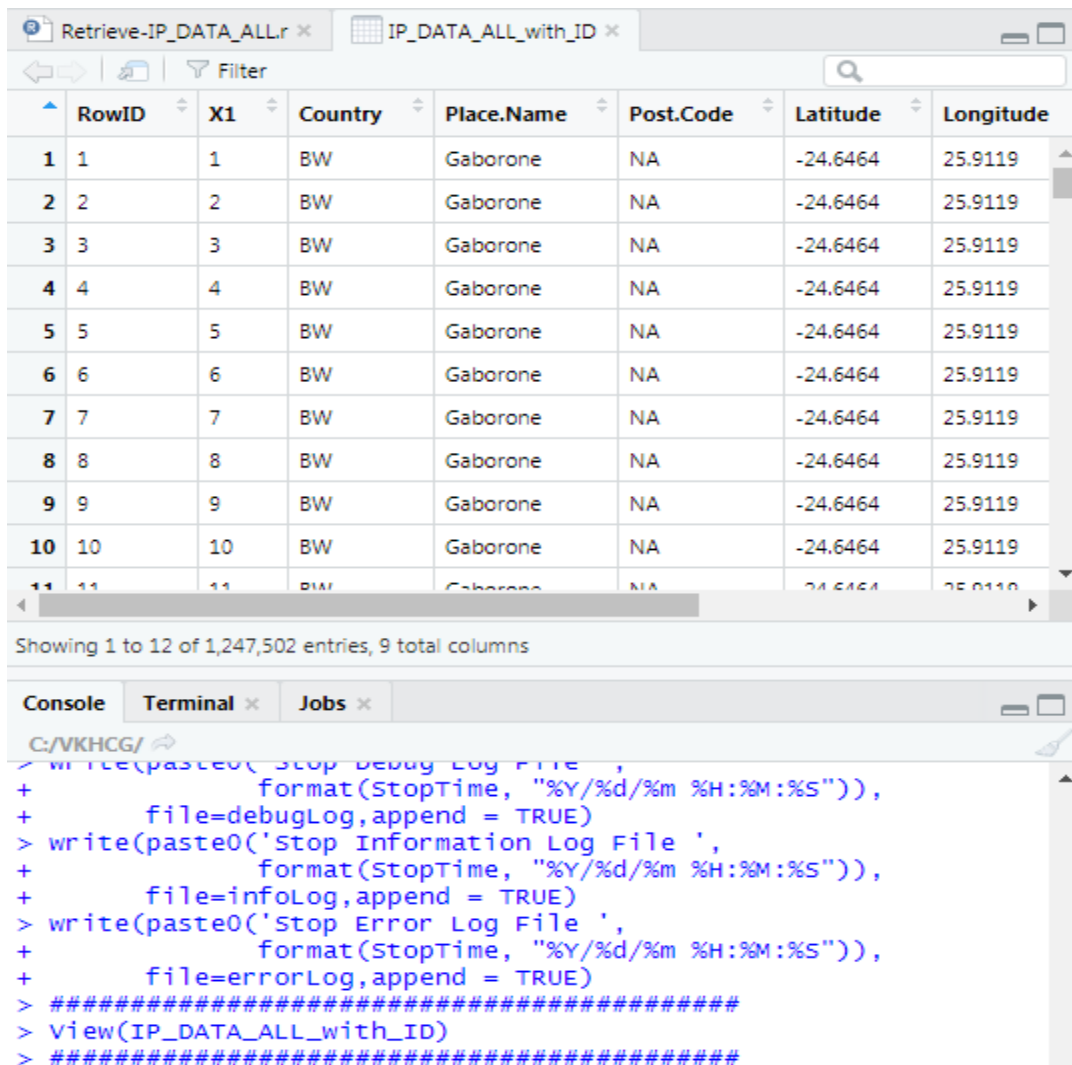
for sCompany in sCompanies:
    sFileDir=Base + '/' + sCompany
    if not os.path.exists(sFileDir):
        os.makedirs(sFileDir)
    for sLayer in sLayers:
        log = logging.getLogger() # root logger
        for hdlr in log.handlers[:]: # remove all old handlers
            log.removeHandler(hdlr)
        sFileDir=Base + '/' + sCompany + '/' + sLayer + '/Logging'
        if os.path.exists(sFileDir):
            shutil.rmtree(sFileDir)
            time.sleep(2)
        if not os.path.exists(sFileDir):
            os.makedirs(sFileDir)
        skey=str(uuid.uuid4())
        sLogFile=Base + '/' + sCompany + '/' + sLayer + '/Logging/'
        Logging_+'skey+'.log'
        print('Set up:',sLogFile)
        logging.basicConfig(level=logging.DEBUG,
            format='%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
            datefmt='%m-%d %H:%M',
            filename=sLogFile,
            filemode='w')
        console = logging.StreamHandler()
        console.setLevel(logging.INFO)
        formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
        console.setFormatter(formatter)
        logging.getLogger("").addHandler(console)
```

```

logging.info('Practical Data Science is fun!.')
for sLevel in sLevels:
sApp='Application-'+ sCompany + '-' + sLayer + '-' + sLevel
logger = logging.getLogger(sApp)
if sLevel == 'debug':
logger.debug('Practical Data Science logged a debugging message.')
if sLevel == 'info':
logger.info('Practical Data Science logged information message.')
if sLevel == 'warning':
logger.warning('Practical Data Science logged a warning message.')
if sLevel == 'error':
logger.error('Practical Data Science logged an error message.')

```

Output:



	RowID	X1	Country	Place.Name	Post.Code	Latitude	Longitude
1	1	1	BW	Gaborone	NA	-24.6464	25.9119
2	2	2	BW	Gaborone	NA	-24.6464	25.9119
3	3	3	BW	Gaborone	NA	-24.6464	25.9119
4	4	4	BW	Gaborone	NA	-24.6464	25.9119
5	5	5	BW	Gaborone	NA	-24.6464	25.9119
6	6	6	BW	Gaborone	NA	-24.6464	25.9119
7	7	7	BW	Gaborone	NA	-24.6464	25.9119
8	8	8	BW	Gaborone	NA	-24.6464	25.9119
9	9	9	BW	Gaborone	NA	-24.6464	25.9119
10	10	10	BW	Gaborone	NA	-24.6464	25.9119
11	11	11	BW	Gaborone	NA	-24.6464	25.9119

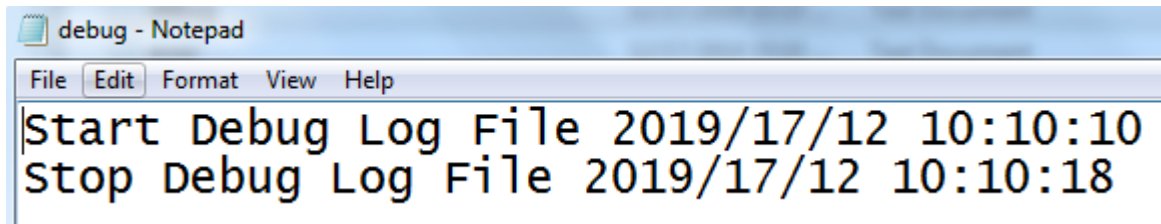
Showing 1 to 12 of 1,247,502 entries, 9 total columns

```

C:\VKHCG\
> write(paste0('Stop Debug Log File ',
+             format(StopTime, "%Y/%d/%m %H:%M:%S")),
+       file=debugLog,append = TRUE)
> write(paste0('Stop Information Log File ',
+             format(StopTime, "%Y/%d/%m %H:%M:%S")),
+       file=infoLog,append = TRUE)
> write(paste0('Stop Error Log File ',
+             format(StopTime, "%Y/%d/%m %H:%M:%S")),
+       file=errorLog,append = TRUE)
> #####
> View(IP_DATA_ALL_with_ID)
> #####

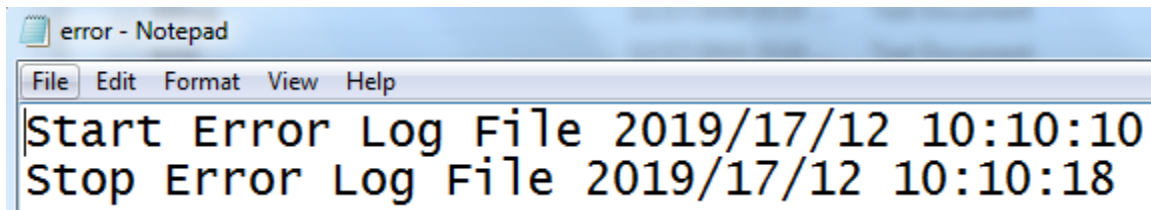
```

Debug.txt



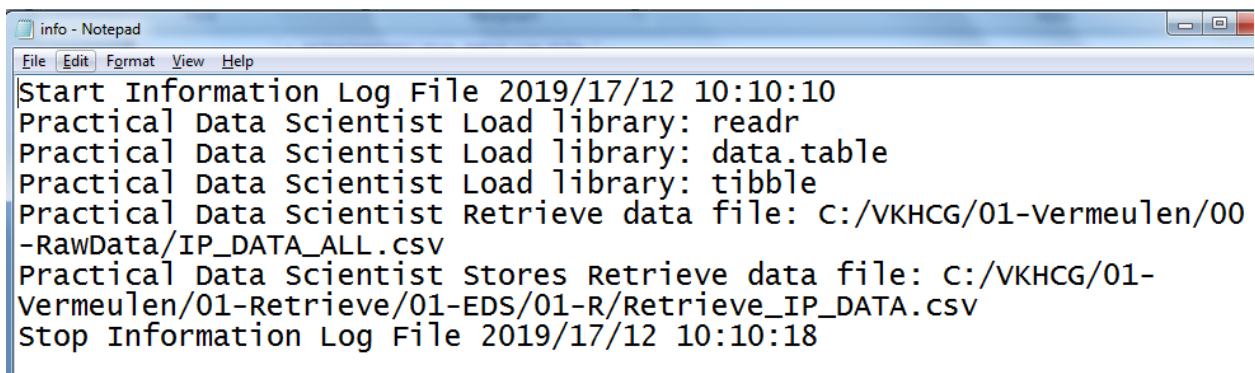
```
debug - Notepad
File Edit Format View Help
Start Debug Log File 2019/17/12 10:10:10
Stop Debug Log File 2019/17/12 10:10:18
```

Error.txt



```
error - Notepad
File Edit Format View Help
Start Error Log File 2019/17/12 10:10:10
Stop Error Log File 2019/17/12 10:10:18
```

Info.txt



```
info - Notepad
File Edit Format View Help
Start Information Log File 2019/17/12 10:10:10
Practical Data Scientist Load library: readr
Practical Data Scientist Load library: data.table
Practical Data Scientist Load library: tibble
Practical Data Scientist Retrieve data file: C:/VKHCG/01-vermeulen/00
-RawData/IP_DATA_ALL.csv
Practical Data Scientist Stores Retrieve data file: C:/VKHCG/01-
Vermeulen/01-Retrieve/01-EDS/01-R/Retrieve_IP_DATA.csv
Stop Information Log File 2019/17/12 10:10:18
```

Practical 4

Retrieve Superstep

A. Perform the following data processing using R.

Code:

```
library(readr)
IP_DATA_ALL <- read_csv("C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv")
View(IP_DATA_ALL)
spec(IP_DATA_ALL)
IP_DATA_ALL_FIX=set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = TRUE)
library(tibble)
set_tidy_names(IP_DATA_ALL, syntactic = TRUE, quiet = FALSE)
sapply(IP_DATA_ALL_FIX, typeof)
library(data.table)
hist_country=data.table(Country=unique(IP_DATA_ALL_FIX[is.na(IP_DATA_ALL_FIX
['Country']) == 0, ]$Country
))
IP_DATA_COUNTRY_FREQ=data.table(with(IP_DATA_ALL_FIX, table(Country)))
View(IP_DATA_COUNTRY_FREQ)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], min, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX[, 'Country'], min, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX[, 'Latitude'], max, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX[, 'Country'], max, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [, 'Latitude'], mean, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [, 'Latitude'], median, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [, 'Latitude'], range, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [, 'Latitude'], quantile, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [, 'Latitude'], sd, na.rm=TRUE)
sapply(IP_DATA_ALL_FIX [, 'Longitude'], sd, na.rm=TRUE)
```

Output:

The screenshot displays the RStudio environment with the following components:

- Environment Panel:** Lists loaded data objects:
 - `hist_country`: 240 obs. of 1 variable
 - `IP_DATA_ALL`: 1247502 obs. of 9 variables
 - `IP_DATA_ALL_...`: 1247502 obs. of 9 variables
 - `IP_DATA_COUN...`: 240 obs. of 2 variables
- Files Panel:** Shows a file explorer view of the home directory with various files and folders.
- Console:** Contains the following R code and output:


```
> sapply(IP_DATA_ALL_FIX [, 'Latitude'], sd, na.rm=TRUE)
Latitude
16.85403
> sapply(IP_DATA_ALL_FIX [, 'Longitude'], sd, na.rm=TRUE)
Longitude
71.78892
```
- Data Viewer:** Displays a table with columns 'Country' and 'N'. The first 11 rows are visible:

	Country	N
1	AD	46
2	AE	1793
3	AF	15
4	AG	21
5	AI	9
6	AL	91
7	AM	92
8	AO	108
9	AR	120
10	AS	5
11	AT	5049

B. Retrive_IP_DATA_ALL.py

Code:

```
import sys
import os
import pandas as pd
#####
Base='C:/VKHCG'
#####
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#####
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
print('Rows:', IP_DATA_ALL.shape[0])
print('Columns:', IP_DATA_ALL.shape[1])
print('### Raw Data Set #####')
for i in range(0,len(IP_DATA_ALL.columns)):
print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('### Fixed Data Set #####')
IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
cNameOld=IP_DATA_ALL_FIX.columns[i] + ' '
cNameNew=cNameOld.strip().replace(" ", ".")
IP_DATA_ALL_FIX.columns.values[i] = cNameNew
print(IP_DATA_ALL_FIX.columns[i],type(IP_DATA_ALL_FIX.columns[i]))
#####
#print(IP_DATA_ALL_FIX.head())
#####
print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ['RowID']
#print(IP_DATA_ALL_with_ID.head())
sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
#####
print('### Done!! #####')
```


Output:

```
In [42]: runfile('C:/Users/MSKIT/.spyder-py3/temp.py',  
wdir='C:/Users/MSKIT/.spyder-py3')  
  
Loading : C:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv  
### Raw Data Set #####  
Unnamed: 0 <class 'str'>  
ID <class 'str'>  
Country <class 'str'>  
Place.Name <class 'str'>  
Post.Code <class 'str'>  
Latitude <class 'str'>  
Longitude <class 'str'>  
First.IP.Number <class 'str'>  
Last.IP.Number <class 'str'>  
### Fixed Data Set #####  
Unnamed:.0 <class 'str'>  
ID <class 'str'>  
Country <class 'str'>  
Place.Name <class 'str'>  
Post.Code <class 'str'>  
Latitude <class 'str'>  
Longitude <class 'str'>  
First.IP.Number <class 'str'>  
Last.IP.Number <class 'str'>
```

C. Data Pattern

Code:

```
library(readr)
library(data.table)
FileName=paste0('c:/VKHCG/01-Vermeulen/00-RawData/IP_DATA_ALL.csv')
IP_DATA_ALL <- read_csv(FileName)
hist_country=data.table(Country=unique(IP_DATA_ALL$Country))
pattern_country=data.table(Country=hist_country$Country,
PatternCountry=hist_country$Country)
oldchar=c(letters,LETTERS)
newchar=replicate(length(oldchar),"A")
for (r in seq(nrow(pattern_country))){
s=pattern_country[r,]$PatternCountry;
for (c in seq(length(oldchar))){
s=chartr(oldchar[c],newchar[c],s)
};
for (n in seq(0,9,1)){
s=chartr(as.character(n),"N",s)
};
s=chartr(" ","b",s)
s=chartr(".", "u",s)
pattern_country[r,]$PatternCountry=s;
};
View(pattern_country)
```

Output:

The screenshot displays the RStudio interface with the following components:

- Environment Pane:** Shows the 'pattern_country' object with 241 rows and 2 variables. The 'Values' section lists:

c	52L
FileName	"c:/VKHCG/01-Vermeulen/00-RawData/IP..."
n	9
newchar	chr [1:52] "A" "A" "A" "A" "A" "A" "A" ...
oldchar	chr [1:52] "a" "b" "c" "d" "e" "f" "f" ...
r	241L
s	"AA"
- Files Pane:** Displays a file explorer view of the 'Home' directory, listing files such as '.Rhistory', 'chisq.R', 'CM Solutions - Unit 1.docx', 'CM Unit I - 11.docx', 'ES practicals syit62.docx', 'Fax', 'GIS DataBase', 'main file stealth.txt', 'Practical 4a.docx', 'Python Scripts', 'R', 'Scanned Documents', 'Virtual Machines', and 'Visual Studio 2008'.
- Console:** Shows the execution of the following R code:


```

s=pattern_country[,1:2]
for (c in seq(length(oldchar))) {
  s=chartr(oldchar[c],newchar[c],s)
}
for (n in seq(0,9,1)) {
  s=chartr(as.character(n),"N",s)
}
s=chartr(" ", "b", s)
s=chartr(".", "u", s)
pattern_country[,1]$PatternCountry=s
}
> view(pattern_country)
      
```

D. Loading IP_DATA_ALL

Code:

```
#####Retrieve-IP_DATA_ALL.py#####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
#####
Base='C:/VKHCG'
#####
sFileName=Base + '/01-Vermeulen/00-RawData/IP_DATA_ALL.csv'
print('Loading :',sFileName)
IP_DATA_ALL=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
#####
sFileDir=Base + '/01-Vermeulen/01-Retrieve/01-EDS/02-Python'
if not os.path.exists(sFileDir):
os.makedirs(sFileDir)
print('Rows:', IP_DATA_ALL.shape[0])
print('Columns:', IP_DATA_ALL.shape[1])
print('### Raw Data Set #####')
for i in range(0,len(IP_DATA_ALL.columns)):
print(IP_DATA_ALL.columns[i],type(IP_DATA_ALL.columns[i]))
print('### Fixed Data Set #####')
IP_DATA_ALL_FIX=IP_DATA_ALL
for i in range(0,len(IP_DATA_ALL.columns)):
cNameOld=IP_DATA_ALL_FIX.columns[i] + ''
cNameNew=cNameOld.strip().replace(" ", ".")
IP_DATA_ALL_FIX.columns.values[i] = cNameNew
print(IP_DATA_ALL_FIX.columns[i],type(IP_DATA_ALL_FIX.columns[i]))
#####
#print(IP_DATA_ALL_FIX.head())
#####
print('Fixed Data Set with ID')
IP_DATA_ALL_with_ID=IP_DATA_ALL_FIX
IP_DATA_ALL_with_ID.index.names = ['RowID']
#print(IP_DATA_ALL_with_ID.head())
sFileName2=sFileDir + '/Retrieve_IP_DATA.csv'
IP_DATA_ALL_with_ID.to_csv(sFileName2, index = True, encoding="latin-1")
#####
print('### Done!! #####')
#####
```

Output:

```
In [59]: runfile('C:/Users/MSKIT/.spyder-py3/temp.py', wdir='C:/
Users/MSKIT/.spyder-py3')

Loading : C:/VKHCG/01-Vermeulen/00-RawData/COUNTRY-CODES.csv
### Raw Data Set #####
Code <class 'str'>
Country name <class 'str'>
Year <class 'str'>
country code top-level domain <class 'str'>
### Fixed Data Set #####
Code <class 'str'>
Country.name <class 'str'>
Year <class 'str'>
country.code.top-level.domain <class 'str'>
```

E. Program to connect to different data sources.

SQLite:

Code:

```
#####
# -*- coding: utf-8 -*-
#####
import sqlite3 as sq
import pandas as pd
#####
Base='C:/VKHCG'
sDatabaseName=Base + '/01-Vermeulen/00-RawData/SQLite/vermeulen.db'
conn = sq.connect(sDatabaseName)
#####
sFileName='C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv'
print('Loading :',sFileName)
IP_DATA_ALL_FIX=pd.read_csv(sFileName,header=0,low_memory=False)
IP_DATA_ALL_FIX.index.names = ['RowIDCSV']
sTable='IP_DATA_ALL'
print('Storing :',sDatabaseName,' Table:',sTable)
IP_DATA_ALL_FIX.to_sql(sTable, conn, if_exists="replace")
print('Loading :',sDatabaseName,' Table:',sTable)
TestData=pd.read_sql_query("select * from IP_DATA_ALL;", conn)
print('#####')
print('## Data Values')
print('#####')
print(TestData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('#####')
print('### Done!! #####')
```

Output:

```

In [9]: runfile('C:/Users/MSCIT/.spyder-py3/temp.py', wdir='C:/Users/
MSCIT/.spyder-py3')

Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/Retrieve_IP_DATA.csv
Storing : C:/VKHCG/01-Vermeulen/00-RowData/SQLite/vermeulen.db Table: IP_DATA_ALL
Loading : C:/VKHCG/01-Vermeulen/00-RowData/SQLite/vermeulen.db Table: IP_DATA_ALL
Data Values
#####

```

	RowIDCSV	RowID	ID	...	Longitude	First.IP.Number	Last.IP.Number
0	0	0	1	...	25.9119	692781056	692781567
1	1	1	2	...	25.9119	692781824	692783103
2	2	2	3	...	25.9119	692909056	692909311
3	3	3	4	...	25.9119	692909568	692910079
4	4	4	5	...	25.9119	693051392	693052415
5	5	5	6	...	25.9119	693078272	693078527
6	6	6	7	...	25.9119	693608448	693616639
7	7	7	8	...	25.9119	696929792	696930047
8	8	8	9	...	25.9119	700438784	700439039
9	9	9	10	...	25.9119	702075904	702076927
10	10	10	11	...	25.9119	702498816	702499839
11	11	11	12	...	25.9119	702516224	702517247
12	12	12	13	...	25.9119	774162663	774162667
13	13	13	14	...	25.9119	1401887232	1401887743
14	14	14	15	...	25.9119	1754209024	1754209279
15	15	15	16	...	2.1167	696918528	696919039
16	16	16	17	...	2.1167	696922112	696924159
17	17	17	18	...	2.1167	701203456	701203711
18	18	18	19	...	2.1167	758886912	758887167
19	19	19	20	...	2.1167	1347294153	1347294160
20	20	20	21	...	2.1167	1755108096	1755108351
21	21	21	22	...	2.1167	1755828480	1755828735
22	22	22	23	...	32.5892	692883456	692883967
23	23	23	24	...	32.5892	692944896	692944943
24	24	24	25	...	32.5892	696967168	696971263
25	25	25	26	...	32.5892	700358656	700360959
26	26	26	27	...	32.5892	702292992	702294015

Practical 5

Assessing Data

5A1. Drop the Columns Where All Elements Are Missing Values

Code:

```
##### Assess-Good-Bad-01.py#####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
#####
Base='C:/VKHCG'
#####
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-01.csv'
Company='01-Vermeulen'
#####
Base='C:/VKHCG'
#####
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####
### Import Warehouse
#####
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('#####')
print('## Raw Data Values')
print('#####')
print(RawData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('#####')
```



```
#####
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
#####
TestData=RawData.dropna(axis=1, how='all')
#####
print('#####')
print('## Test Data Values')
print('#####')
print(TestData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows :',TestData.shape[0])
print('Columns :',TestData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#####
print('#####')
print('### Done!! #####')
print('#####')
```

Output:

```
In [16]: runfile('C:/VKHCG/01-Vermeulen/02-Assess/Assess-Good-
Rad-01.py', wdir='C:/VKHCG/01-Vermeulen/02-Assess')
```

```
#####
Working Base : C:/VKHCG using win32
#####
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
#####
## Raw Data Values
#####
   ID FieldA FieldB FieldC FieldD FieldE FieldF FieldG
0   1.0   Good   Better   Best 1024.0   NaN 10241.0    1
1   2.0   Good    NaN   Best  512.0   NaN  5121.0    2
2   3.0   Good   Better   NaN  256.0   NaN  256.0    3
3   4.0   Good   Better   Best   NaN   NaN  211.0    4
4   5.0   Good   Better   NaN  64.0   NaN 6411.0    5
5   6.0   Good    NaN   Best  32.0   NaN  32.0    6
6   7.0   NaN   Better   Best  16.0   NaN 1611.0    7
7   8.0   NaN    NaN   Best   8.0   NaN 8111.0    8
8   9.0   NaN    NaN   NaN    4.0   NaN  41.0    9
9  10.0    A     B     C     2.0   NaN 21111.0   10
10  NaN    NaN    NaN    NaN    NaN   NaN   NaN    11
11 10.0   Good   Better   Best 1024.0   NaN 102411.0   12
12 10.0   Good    NaN   Best  512.0   NaN  512.0    13
13 10.0   Good   Better   NaN  256.0   NaN 1256.0    14
14 10.0   Good   Better   Best   NaN   NaN   NaN    15
15 10.0   Good   Better   NaN  64.0   NaN  164.0    16
16 10.0   Good    NaN   Best  32.0   NaN  322.0    17
17 10.0   NaN   Better   Best  16.0   NaN  163.0    18
18 10.0   NaN    NaN   Best   8.0   NaN  844.0    19
19 10.0   NaN    NaN   NaN    4.0   NaN 4555.0    20
20 10.0    A     B     C     2.0   NaN  111.0    21

#####
## Data Profile
#####
Rows : 21
Columns : 8
#####
## Test Data Values
#####
   ID FieldA FieldB FieldC FieldD FieldE FieldF FieldG
0   1.0   Good   Better   Best 1024.0 10241.0    1
1   2.0   Good    NaN   Best  512.0  5121.0    2
2   3.0   Good   Better   NaN  256.0  256.0    3
3   4.0   Good   Better   Best   NaN  211.0    4
4   5.0   Good   Better   NaN  64.0 6411.0    5
5   6.0   Good    NaN   Best  32.0  32.0    6
6   7.0   NaN   Better   Best  16.0 1611.0    7
7   8.0   NaN    NaN   Best   8.0 8111.0    8
8   9.0   NaN    NaN   NaN    4.0  41.0    9
9  10.0    A     B     C     2.0 21111.0   10
10  NaN    NaN    NaN    NaN    NaN   NaN    11
11 10.0   Good   Better   Best 1024.0 102411.0   12
12 10.0   Good    NaN   Best  512.0  512.0    13
13 10.0   Good   Better   NaN  256.0 1256.0    14
14 10.0   Good   Better   Best   NaN   NaN    15
15 10.0   Good   Better   NaN  64.0  164.0    16
16 10.0   Good    NaN   Best  32.0  322.0    17
17 10.0   NaN   Better   Best  16.0  163.0    18
18 10.0   NaN    NaN   Best   8.0  844.0    19
19 10.0   NaN    NaN   NaN    4.0 4555.0    20
20 10.0    A     B     C     2.0  111.0    21

#####
## Data Profile
#####
Rows : 21
Columns : 7
#####
Done!!
```

5A2. Drop the Columns Where Any of the Elements Is Missing Values

Code:

```
##### Assess-Good-Bad-02.py#####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
#####
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-02.csv'
Company='01-Vermeulen'
#####
Base='C:/VKHCG'
#####
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####
### Import Warehouse
#####
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('#####')
print('## Raw Data Values')
print('#####')
print(RawData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
#####
TestData=RawData.dropna(axis=1, how='any')
```

```
#####
print('#####')
print('## Test Data Values')
print('#####')
print(TestData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows :', TestData.shape[0])
print('Columns :', TestData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#####
print('#####')
print('### Done!! #####')
print('#####')
```

Output:

```
In [22]: runfile('C:/VKHCG/01-Vermeulen/02-Assess/Assess-Good-Bad-02.py', wdir='C:/VKHCG/01-Vermeulen/02-Assess')
```

```
#####
Working Base : C:/VKHCG using win32
#####
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
#####
## Raw Data Values
#####
      ID FieldA  FieldB FieldC FieldD FieldE FieldF FieldG
0    1.0   Good  Better   Best  1024.0   NaN  10241.0    1
1    2.0   Good   NaN    Best   512.0   NaN   5121.0    2
2    3.0   Good  Better   NaN   256.0   NaN   256.0    3
3    4.0   Good  Better   Best    NaN   NaN   211.0    4
4    5.0   Good  Better   NaN    64.0   NaN   6411.0    5
5    6.0   Good   NaN    Best    32.0   NaN    32.0    6
6    7.0   NaN   Better   Best    16.0   NaN   1611.0    7
7    8.0   NaN   NaN    Best     8.0   NaN   8111.0    8
8    9.0   NaN   NaN    NaN     4.0   NaN    41.0    9
9   10.0    A     B     C     2.0   NaN  21111.0   10
10   NaN   NaN   NaN    NaN    NaN   NaN    NaN    11
11  10.0   Good  Better   Best  1024.0   NaN  102411.0   12
12  10.0   Good   NaN    Best   512.0   NaN   512.0   13
13  10.0   Good  Better   NaN   256.0   NaN  1256.0   14
14  10.0   Good  Better   Best    NaN   NaN    NaN    15
15  10.0   Good  Better   NaN    64.0   NaN   164.0   16
16  10.0   Good   NaN    Best    32.0   NaN   322.0   17
17  10.0   NaN   Better   Best    16.0   NaN   163.0   18
18  10.0   NaN   NaN    Best     8.0   NaN   844.0   19
19  10.0   NaN   NaN    NaN     4.0   NaN  4555.0   20
20  10.0    A     B     C     2.0   NaN   111.0   21
```

```
#####
## Data Profile
#####
Rows : 21
Columns : 8
#####
## Test Data Values
#####
      FieldG
0          1
1          2
2          3
3          4
4          5
5          6
6          7
7          8
8          9
9         10
10         11
11         12
12         13
13         14
14         15
15         16
16         17
17         18
18         19
19         20
20         21
#####
## Data Profile
#####
Rows : 21
Columns : 1
#####
Done!!
```

5A3. Keep Only the Rows That Contain a Maximum of Two Missing Values

Code:

```
##### Assess-Good-Bad-03.py #####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
#####
sInputFileName='Good-or-Bad.csv'
sOutputFileName='Good-or-Bad-03.csv'
Company='01-Vermeulen'
Base='C:/VKHCG'
#####
print('#####')
print('Working Base :',Base, ' using Windows ~~~~')
print('#####')
#####
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####
### Import Warehouse
#####
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading :',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('#####')
print('## Raw Data Values')
print('#####')
print(RawData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows :',RawData.shape[0])
print('Columns :',RawData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sInputFileName
RawData.to_csv(sFileName, index = False)
#####
TestData=RawData.dropna(thresh=2)

print('#####')
```

```

print('## Test Data Values')
print('#####')
print(TestData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows :', TestData.shape[0])
print('Columns :', TestData.shape[1])
print('#####')
sFileName=sFileDir + '/' + sOutputFileName
TestData.to_csv(sFileName, index = False)
#####
print('#####')
print('### Done!! #####')
print('#####')

```

Output:

```

In [35]: runfile('C:/VKHCG/01-Vermeulen/02-Assess/Assess-Good-
Bad-03.py', wdir='C:/VKHCG/01-Vermeulen/02-Assess')

#####
Working Base : C:/VKHCG using win32
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
#####
## Raw Data Values
#####
   ID FieldA FieldB FieldC FieldD FieldE FieldF FieldG
0  1.0   Good  Better   Best  1024.0   NaN  10241.0    1
1  2.0   Good   NaN    Best   512.0   NaN   5121.0    2
2  3.0   Good  Better   NaN   256.0   NaN   256.0    3
3  4.0   Good  Better   Best    NaN    NaN   211.0    4
4  5.0   Good  Better   NaN    64.0   NaN   6411.0    5
5  6.0   Good   NaN    Best    32.0   NaN    32.0    6
6  7.0   NaN   Better   Best    16.0   NaN   1611.0    7
7  8.0   NaN   NaN     Best     8.0   NaN   8111.0    8
8  9.0   NaN   NaN     NaN     4.0   NaN    41.0    9
9 10.0    A     B       C       2.0   NaN  21111.0   10
10 NaN    NaN   NaN     NaN     NaN   NaN    NaN     11
11 10.0   Good  Better   Best  1024.0   NaN  102411.0  12
12 10.0   Good   NaN    Best   512.0   NaN   512.0    13
13 10.0   Good  Better   NaN   256.0   NaN  1256.0    14
14 10.0   Good  Better   Best    NaN   NaN    NaN     15
15 10.0   Good  Better   NaN    64.0   NaN   164.0    16
16 10.0   Good   NaN    Best    32.0   NaN   322.0    17
17 10.0   NaN   Better   Best    16.0   NaN   163.0    18
18 10.0   NaN   NaN     Best     8.0   NaN   844.0    19
19 10.0   NaN   NaN     NaN     4.0   NaN  4555.0    20
20 10.0    A     B       C       2.0   NaN   111.0    21
#####
## Data Profile
#####
Rows : 21
Columns : 8

```

```

## Test Data Values
#####
   ID FieldA FieldB FieldC FieldD FieldE FieldF FieldG
0  1.0   Good  Better   Best  1024.0   NaN  10241.0    1
1  2.0   Good   NaN    Best   512.0   NaN   5121.0    2
2  3.0   Good  Better   NaN   256.0   NaN   256.0    3
3  4.0   Good  Better   Best    NaN    NaN   211.0    4
4  5.0   Good  Better   NaN    64.0   NaN   6411.0    5
5  6.0   Good   NaN    Best    32.0   NaN    32.0    6
6  7.0   NaN   Better   Best    16.0   NaN   1611.0    7
7  8.0   NaN   NaN     Best     8.0   NaN   8111.0    8
8  9.0   NaN   NaN     NaN     4.0   NaN    41.0    9
9 10.0    A     B       C       2.0   NaN  21111.0   10
11 10.0   Good  Better   Best  1024.0   NaN  102411.0  12
12 10.0   Good   NaN    Best   512.0   NaN   512.0    13
13 10.0   Good  Better   NaN   256.0   NaN  1256.0    14
14 10.0   Good  Better   Best    NaN   NaN    NaN     15
15 10.0   Good  Better   NaN    64.0   NaN   164.0    16
16 10.0   Good   NaN    Best    32.0   NaN   322.0    17
17 10.0   NaN   Better   Best    16.0   NaN   163.0    18
18 10.0   NaN   NaN     Best     8.0   NaN   844.0    19
19 10.0   NaN   NaN     NaN     4.0   NaN  4555.0    20
20 10.0    A     B       C       2.0   NaN   111.0    21
#####
## Data Profile
#####
Rows : 20
Columns : 8
#####
Done!!

```

5A4. Fill All Missing Values with the Mean, Median, Mode, Minimum, and Maximum of the Particular Numeric Column

Code:

```
# -*- coding: utf-8 -
##### Import sys
import os
import pandas as pd
#####
Base='C:/VKHCG'
sInputFileName='Good-or-Bad.csv'
sOutputFileNameA='Good-or-Bad-04-A.csv'
sOutputFileNameB='Good-or-Bad-04-B.csv'
sOutputFileNameC='Good-or-Bad-04-C.csv'
sOutputFileNameD='Good-or-Bad-04-D.csv'
sOutputFileNameE='Good-or-Bad-04-E.csv'
Company='01-Vermeulen'
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
    print('#####')
    print('Working Base:',Base,'using',sys.platform)
    print('#####')
    #####
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
    #####
    ### Import Warehouse
    #####
sFileName=Base + '/' + Company + '/00-RawData/' + sInputFileName
print('Loading:',sFileName)
RawData=pd.read_csv(sFileName,header=0)
print('#####')
print('## Raw Data Values')
print('#####')
print(RawData) print('#####')
print('## Data Profile')
print('#####')
print('Rows:',RawData.shape[0])
print('Columns:',RawData.shape[1])
print('#####')
    #####
sFileName=sFileDir + '/' + sInputFileName
```

```

RawData.to_csv(sFileName,index=False)
#####
TestData=RawData.fillna(RawData.mean())
#####
print('#####')
print('## Test Data Values- Mean')
print('#####')
print(TestData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows:',TestData.shape[0])
print('Columns:',TestData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sOutputFileNameA
TestData.to_csv(sFileName,index=False)

#####
## This is the important action! The rest of this code snippet
## Only supports this action.
#####
TestData=RawData.fillna(RawData.median())
#####
print('#####')
print('## Test Data Values - Median')
print('#####')
print(TestData)
print('#####')
print('## Data Profile')
print('#####')
print('Rows:',TestData.shape[0])
print('Columns:',TestData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sOutputFileNameB
TestData.to_csv(sFileName,index=False)
#####
#####
TestData=RawData.fillna(RawData.mode())
#####
print('#####')
print('## Test Data Values - Mode')
print('#####')
print(TestData)
print('#####')

```



```

print('## Data Profile')
print('#####')
print('Rows:', TestData.shape[0])
print('Columns:', TestData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sOutputFileNameC
TestData.to_csv(sFileName,index=False)
#####
#####
TestData=RawData.fillna(RawData.min())
#####
print('#####')
print('## Test Data Values - Minumum')
print('#####')
print(TestData) print('#####')
print('## Data Profile')
print('#####')
print('Rows:', TestData.shape[0])
print('Columns:', TestData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sOutputFileNameD
TestData.to_csv(sFileName,index=False)
#####
#####
TestData=RawData.fillna(RawData.max())
#####
print('#####')
print('## Test Data Values - Maximum')
print('#####')
print(TestData)
print('#####')
print('##DataProfile')
print('#####')
print('Rows:', TestData.shape[0])
print('Columns:', TestData.shape[1])
print('#####')
#####
sFileName=sFileDir + '/' + sOutputFileNameE
TestData.to_csv(sFileName,index=False)
#####
#####
print('#####')
print('### Done!! #####')

```

Output:

```
In [7]: runfile('C:/VKHCG/01-Vermeulen/02-Assess/Assess-Good-Bad-05.py',
wdir='C:/VKHCG/01-Vermeulen/02-Assess')
```

```
#####
Working Base : C:/VKHCG using win32
#####
Loading : C:/VKHCG/01-Vermeulen/00-RawData/Good-or-Bad.csv
#####
## Raw Data Values
#####
      ID FieldA FieldB FieldC FieldD FieldE FieldF FieldG
0    1.0   Good  Better   Best  1024.0   NaN  10241.0     1
1    2.0   Good    NaN   Best   512.0   NaN   5121.0     2
2    3.0   Good  Better    NaN   256.0   NaN    256.0     3
3    4.0   Good  Better   Best    NaN   NaN    211.0     4
4    5.0   Good  Better    NaN    64.0   NaN   6411.0     5
5    6.0   Good    NaN   Best    32.0   NaN    32.0     6
6    7.0   NaN  Better   Best    16.0   NaN   1611.0     7
7    8.0   NaN    NaN   Best     8.0   NaN   8111.0     8
8    9.0   NaN    NaN    NaN     4.0   NaN    41.0     9
9   10.0     A     B     C     2.0   NaN  21111.0    10
10   NaN    NaN    NaN    NaN    NaN   NaN     NaN    11
11  10.0   Good  Better   Best  1024.0   NaN  102411.0    12
12  10.0   Good    NaN   Best   512.0   NaN    512.0    13
13  10.0   Good  Better    NaN   256.0   NaN   1256.0    14
14  10.0   Good  Better   Best    NaN   NaN     NaN    15
15  10.0   Good  Better    NaN    64.0   NaN    164.0    16
16  10.0   Good    NaN   Best    32.0   NaN    322.0    17
17  10.0   NaN  Better   Best    16.0   NaN    163.0    18
18  10.0   NaN    NaN   Best     8.0   NaN    844.0    19
19  10.0   NaN    NaN    NaN     4.0   NaN   4555.0    20
20  10.0     A     B     C     2.0   NaN    111.0    21
#####
## Data Profile
```

Practical 6

Processing Data Hubs, Links, Satellite

```
#####
# -*- coding: utf-8 -*-
#####
import sys
import os
from datetime import datetime
from datetime import timedelta
from pytz import timezone, all_timezones
import pandas as pd
import sqlite3 as sq
from pandas.io import sql
import uuid

pd.options.mode.chained_assignment = None
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
Company='01-Vermeulen'
InputDir='00-RawData'
InputFileName='VehicleData.csv'
#####
sDataBaseDir=Base + '/' + Company + '/03-Process/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#####
sDatabaseName=sDataBaseDir + '/Hillman.db'
conn1 = sq.connect(sDatabaseName)
#####
sDataVaultDir=Base + '/88-DV'
if not os.path.exists(sDataVaultDir):
    os.makedirs(sDataVaultDir)
#####
sDatabaseName=sDataVaultDir + '/datavault.db'
conn2 = sq.connect(sDatabaseName)
#####
```

```

base = datetime(2018,1,1,0,0,0)
numUnits=10*365*24
#####
date_list = [base - timedelta(hours=x) for x in range(0, numUnits)]
t=0
for i in date_list:
    now_utc=i.replace(tzinfo=timezone('UTC'))
    sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")
    print(sDateTime)
    sDateTimeKey=sDateTime.replace(' ','-').replace(':', '-')
    t+=1
    IDNumber=str(uuid.uuid4())
    TimeLine=[('ZoneBaseKey', ['UTC']),
              ('IDNumber', [IDNumber]),
              ('nDateTimeValue', [now_utc]),
              ('DateTimeValue', [sDateTime]),
              ('DateTimeKey', [sDateTimeKey])]
    if t==1:
        TimeFrame = pd.DataFrame.from_items(TimeLine)
    else:
        TimeRow = pd.DataFrame.from_items(TimeLine)
        TimeFrame = TimeFrame.append(TimeRow)
#####
TimeHub=TimeFrame[['IDNumber','ZoneBaseKey','DateTimeKey','DateTimeValue']]
TimeHubIndex=TimeHub.set_index(['IDNumber'],inplace=False)
#####
TimeFrame.set_index(['IDNumber'],inplace=True)
#####
sTable = 'Process-Time'
print('Storing :',sDatabaseName,' Table:',sTable)
TimeHubIndex.to_sql(sTable, conn1, if_exists="replace")
#####
sTable = 'Hub-Time'
print('Storing :',sDatabaseName,' Table:',sTable)
TimeHubIndex.to_sql(sTable, conn2, if_exists="replace")
#####
active_timezones=all_timezones
z=0
for zone in active_timezones:
    t=0
    for j in range(TimeFrame.shape[0]):
        now_date=TimeFrame['nDateTimeValue'][j]
        DateTimeKey=TimeFrame['DateTimeKey'][j]
        now_utc=now_date.replace(tzinfo=timezone('UTC'))
        sDateTime=now_utc.strftime("%Y-%m-%d %H:%M:%S")
        now_zone = now_utc.astimezone(timezone(zone))

```

```

sZoneDateTime=now_zone.strftime("%Y-%m-%d %H:%M:%S")
print(sZoneDateTime)
t+=1
z+=1
IDZoneNumber=str(uuid.uuid4())
TimeZoneLine=[('ZoneBaseKey', ['UTC']),
               ('IDZoneNumber', [IDZoneNumber]),
               ('DateTimeKey', [DateTimeKey]),
               ('UTCDateTimeValue', [sDateTime]),
               ('Zone', [zone]),
               ('DateTimeValue', [sZoneDateTime])]
if t==1:
    TimeZoneFrame = pd.DataFrame.from_items(TimeZoneLine)
else:
    TimeZoneRow = pd.DataFrame.from_items(TimeZoneLine)
    TimeZoneFrame = TimeZoneFrame.append(TimeZoneRow)

TimeZoneFrameIndex=TimeZoneFrame.set_index(['IDZoneNumber'],inplace=False)
sZone=zone.replace('/', '-').replace(' ', '')
#####
sTable = 'Process-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn1, if_exists="replace")
#####
#####
sTable = 'Satellite-Time-'+sZone
print('Storing :',sDatabaseName,' Table:',sTable)
TimeZoneFrameIndex.to_sql(sTable, conn2, if_exists="replace")
#####
print('#####')
print('Vacuum Databases')
sSQL="VACUUM;"
sql.execute(sSQL,conn1)
sql.execute(sSQL,conn2)
print('#####')
#####
print('### Done!! #####')
#####

```

Output:

```
In [33]: runfile('C:/Users/MSCT/.spyder-py3/temp.py',
wdir='C:/Users/MSCT/.spyder-py3')

#####
Working Base : C:/VKHCG using win32
#####
2018-01-01 00:00:00
2017-12-31 23:00:00
2017-12-31 22:00:00
2017-12-31 21:00:00
2017-12-31 20:00:00
2017-12-31 19:00:00
2017-12-31 18:00:00
2017-12-31 17:00:00
2017-12-31 16:00:00
2017-12-31 15:00:00
Storing : C:/VKHCG/88-DV/datavault.db Table: Process-Time

Storing : C:/VKHCG/88-DV/datavault.db Table: Hub-Time
2018-01-01 00:00:00
2017-12-31 23:00:00
2017-12-31 22:00:00
2017-12-31 21:00:00
2017-12-31 20:00:00
2017-12-31 19:00:00
2017-12-31 18:00:00
2017-12-31 17:00:00
2017-12-31 16:00:00
2017-12-31 15:00:00
Storing : C:/VKHCG/88-DV/datavault.db Table: Process-Time-
Africa-Abidjan

Storing : C:/VKHCG/88-DV/datavault.db Table: Satellite-
Time-Africa-Abidjan
2018-01-01 00:00:00
2017-12-31 23:00:00
2017-12-31 22:00:00
2017-12-31 21:00:00
2017-12-31 20:00:00
2017-12-31 19:00:00
2017-12-31 18:00:00
2017-12-31 17:00:00
2017-12-31 16:00:00
2017-12-31 15:00:00
```

Practical 7

Transforming Data

Sun Model

```
#####
# -*- coding: utf-8 -*-
#####
import sys
import os
from datetime import datetime
from pytz import timezone
import pandas as pd
import sqlite3 as sq
import uuid
pd.options.mode.chained_assignment = None
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
Company='01-Vermeulen'
#####
sDataBaseDir=Base + '/' + Company + '/04-Transform/SQLite'
if not os.path.exists(sDataBaseDir):
    os.makedirs(sDataBaseDir)
#####
sDatabaseName=sDataBaseDir + '/Vermeulen.db'
conn1 = sq.connect(sDatabaseName)
#####
sDataWarehousetDir=Base + '/99-DW'
if not os.path.exists(sDataWarehousetDir):
    os.makedirs(sDataWarehousetDir)
#####
sDatabaseName=sDataWarehousetDir + '/datawarehouse.db'
conn2 = sq.connect(sDatabaseName)
#####
print('\n#####')
print('Time Dimension')
BirthZone = 'Atlantic/Reykjavik'
BirthDateUTC = datetime(1960,12,20,10,15,0)
BirthDateZoneUTC=BirthDateUTC.replace(tzinfo=timezone('UTC'))
BirthDateZoneStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S")
BirthDateZoneUTCStr=BirthDateZoneUTC.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
```

```

BirthDate = BirthDateZoneUTC.astimezone(timezone(BirthZone))
BirthDateStr=BirthDate.strftime("%Y-%m-%d %H:%M:%S (%Z) (%z)")
BirthDateLocal=BirthDate.strftime("%Y-%m-%d %H:%M:%S")
#####
IDTimeNumber=str(uuid.uuid4())
TimeLine=[('TimeID', [IDTimeNumber]),
          ('UTCDate', [BirthDateZoneStr]),
          ('LocalTime', [BirthDateLocal]),
          ('TimeZone', [BirthZone])]
TimeFrame = pd.DataFrame.from_items(TimeLine)
#####
DimTime=TimeFrame
DimTimeIndex=DimTime.set_index(['TimeID'],inplace=False)
#####
sTable = 'Dim-Time'
print("\n#####")
print('Storing :',sDatabaseName,'\n Table:',sTable)
print("\n#####")
DimTimeIndex.to_sql(sTable, conn1, if_exists="replace")
DimTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####
print("\n#####")
print('Dimension Person')
print("\n#####")
FirstName = 'Guðmundur'
LastName = 'Gunnarsson'
#####
IDPersonNumber=str(uuid.uuid4())
PersonLine=[('PersonID', [IDPersonNumber]),
            ('FirstName', [FirstName]),
            ('LastName', [LastName]),
            ('Zone', ['UTC']),
            ('DateTimeValue', [BirthDateZoneStr])]
PersonFrame = pd.DataFrame.from_items(PersonLine)
#####
DimPerson=PersonFrame
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####
sTable = 'Dim-Person'
print("\n#####")
print('Storing :',sDatabaseName,'\n Table:',sTable)
print("\n#####")
DimPersonIndex.to_sql(sTable, conn1, if_exists="replace")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####
print("\n#####")

```



```

print('Fact - Person - time')
print("\n#####")
IDFactNumber=str(uuid.uuid4())
PersonTimeLine=[('IDNumber', [IDFactNumber]),
                 ('IDPersonNumber', [IDPersonNumber]),
                 ('IDTimeNumber', [IDTimeNumber])]
PersonTimeFrame = pd.DataFrame.from_items(PersonTimeLine)
#####
FctPersonTime=PersonTimeFrame
FctPersonTimeIndex=FctPersonTime.set_index(['IDNumber'],inplace=False)
#####
sTable = 'Fact-Person-Time'
print("\n#####")
print('Storing :',sDatabaseName,'\n Table:',sTable)
print("\n#####")
FctPersonTimeIndex.to_sql(sTable, conn1, if_exists="replace")
FctPersonTimeIndex.to_sql(sTable, conn2, if_exists="replace")
#####

```

Output:

```
In [24]: runfile('C:/VKHCG/01-Vermeulen/04-Transform/Transform-
Gunnarsson-Sun-Model.py', wdir='C:/VKHCG/01-Vermeulen/04-
Transform')

#####
Working Base : C:/VKHCG using win32
#####

#####
Time Dimension

#####
Storing : C:/VKHCG/99-DW/datawarehouse.db
Table: Dim-Time

#####
C:/VKHCG/01-Vermeulen/04-Transform/Transform-Gunnarsson-Sun-
Model.py:55: FutureWarning: from_items is deprecated. Please use
DataFrame.from_dict(dict(items), ...) instead.
DataFrame.from_dict(OrderedDict(items)) may be used to preserve
the key order.
    DimTime=TimeFrame

#####
Dimension Person

#####

#####
Storing : C:/VKHCG/99-DW/datawarehouse.db
Table: Dim-Person

#####
C:/VKHCG/01-Vermeulen/04-Transform/Transform-Gunnarsson-Sun-
Model.py:79: FutureWarning: from_items is deprecated. Please use
DataFrame.from_dict(dict(items), ...) instead.
DataFrame.from_dict(OrderedDict(items)) may be used to preserve
the key order.
    DimPerson=PersonFrame

#####
Fact - Person - time

#####

#####
Storing : C:/VKHCG/99-DW/datawarehouse.db
Table: Fact-Person-Time

#####
C:/VKHCG/01-Vermeulen/04-Transform/Transform-Gunnarsson-Sun-
Model.py:98: FutureWarning: from_items is deprecated. Please use
DataFrame.from_dict(dict(items), ...) instead.
DataFrame.from_dict(OrderedDict(items)) may be used to preserve
the key order.
    FctPersonTime=PersonTimeFrame
```

Practical 8A Organizing Data

A. Horizontal Style

```
#####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
import sqlite3 as sq
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
#####
Company='01-Vermeulen'
#####
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
#####
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####
print('#####')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####
print('#####')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#####')
sSQL="SELECT PersonID,\
      Height,\
      Weight,\
```

```

        bmi,\
        Indicator\
    FROM [Dim-BMI]\
    WHERE \
    Height > 1.5 \
    and Indicator = 1\
    ORDER BY \
        Height,\
        Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['PersonID'],inplace=False)
#####
sTable = 'Dim-BMI-Horizontal'
print("\n#####")
print('Storing :',sDatabaseName,'\n Table:',sTable)
print("\n#####")
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####
print('#####')
sTable = 'Dim-BMI-Horizontal'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#####')
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####
print('#####')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('#####')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('#####')
#####

```

Output:

```
In [20]: runfile('C:/VKHCG/01-Vermeulen/05-Organise/Organize-Horizontal.py', wdir='C:/VKHCG/01-Vermeulen/05-Organise')
```

```
#####
Working Base : C:/VKHCG using win32
#####
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI
#####

#####
Storing : C:/VKHCG/99-DW/datamart.db
Table: Dim-BMI-Horizontal

#####
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI-Horizontal
#####
#####
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
#####
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
#####
```

Practical 8B

Vertical style

```
#####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
import sqlite3 as sq
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
#####
Company='01-Vermeulen'
#####
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
#####
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####
print('#####')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####
print('#####')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#####')
sSQL="SELECT \
    Height,\
    Weight,\
    Indicator\
FROM [Dim-BMI];"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####
DimPerson=PersonFrame1
```

```

DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####
sTable = 'Dim-BMI-Vertical'
print('\n#####')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#####')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####
print('#####')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####
print('#####')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('#####')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('#####')
#####

```

Output:

```

In [20]: runfile('C:/VKHCG/01-Vermeulen/05-Organise/Organize-Horizontal.py', wdir='C:/VKHCG/
01-Vermeulen/05-Organise')

```

```

#####
Working Base : C:/VKHCG using win32
#####
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI
#####

#####
Storing : C:/VKHCG/99-DW/datamart.db
Table: Dim-BMI-Horizontal

#####
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI-Horizontal
#####
#####
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
#####
Horizontal Data Set (Rows): 194
Horizontal Data Set (Columns): 5
#####

```

Practical 8C

Island Style

```
#####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
import sqlite3 as sq
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
#####
Company='01-Vermeulen'
#####
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
#####
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####
print('#####')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####
print('#####')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
    Height,\
    Weight,\
```



```

Indicator\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
    Height,\
    Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####
sTable = 'Dim-BMI-Vertical'
print('\n#####')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#####')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####
print('#####')
sTable = 'Dim-BMI-Vertical'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#####')
sSQL="SELECT * FROM [Dim-BMI-Vertical];"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####
print('#####')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('#####')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])
print('#####')
#####

```

Output:

```
In [38]: runfile('C:/VKHCG/01-Vermeulen/05-Organise/Organize-Island.py', wdir='C:/VKHCG/01-Vermeulen/05-Organise')
```

```
#####  
Working Base : C:/VKHCG using win32  
#####  
#####  
Loading : C:/VKHCG/99-Dw/datamart.db Table: Dim-BMI  
#####  
Loading : C:/VKHCG/99-Dw/datamart.db Table: Dim-BMI  
  
#####  
Storing : C:/VKHCG/99-Dw/datamart.db  
Table: Dim-BMI-Vertical  
  
#####  
#####  
Loading : C:/VKHCG/99-Dw/datamart.db Table: Dim-BMI-Vertical  
#####  
#####  
Full Data Set (Rows): 1080  
Full Data Set (Columns): 5  
#####  
Horizontal Data Set (Rows): 771  
Horizontal Data Set (Columns): 3  
#####
```

Practical 8D

Secure Vault Style

```
#####
# -*- coding: utf-8 -*-
#####
import sys
import os
import pandas as pd
import sqlite3 as sq
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
#####
Company='01-Vermeulen'
#####
sDataWarehouseDir=Base + '/99-DW'
if not os.path.exists(sDataWarehouseDir):
    os.makedirs(sDataWarehouseDir)
#####
sDatabaseName=sDataWarehouseDir + '/datawarehouse.db'
conn1 = sq.connect(sDatabaseName)
#####
sDatabaseName=sDataWarehouseDir + '/datamart.db'
conn2 = sq.connect(sDatabaseName)
#####
print('#####')
sTable = 'Dim-BMI'
print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT * FROM [Dim-BMI];"
PersonFrame0=pd.read_sql_query(sSQL, conn1)
#####
print('#####')
sTable = 'Dim-BMI'
```

```

print('Loading :',sDatabaseName,' Table:',sTable)
sSQL="SELECT \
    Height,\
    Weight,\
    Indicator,\
    CASE Indicator\
    WHEN 1 THEN 'Pip'\
    WHEN 2 THEN 'Norman'\
    WHEN 3 THEN 'Grant'\
    ELSE 'Sam'\
    END AS Name\
FROM [Dim-BMI]\
WHERE Indicator > 2\
ORDER BY \
    Height,\
    Weight;"
PersonFrame1=pd.read_sql_query(sSQL, conn1)
#####
DimPerson=PersonFrame1
DimPersonIndex=DimPerson.set_index(['Indicator'],inplace=False)
#####
sTable = 'Dim-BMI-Secure'
print('\n#####')
print('Storing :',sDatabaseName,'\n Table:',sTable)
print('\n#####')
DimPersonIndex.to_sql(sTable, conn2, if_exists="replace")
#####
print('#####')
sTable = 'Dim-BMI-Secure'
print('Loading :',sDatabaseName,' Table:',sTable)
print('#####')
sSQL="SELECT * FROM [Dim-BMI-Secure] WHERE Name = 'Sam';"
PersonFrame2=pd.read_sql_query(sSQL, conn2)
#####
print('#####')
print('Full Data Set (Rows):', PersonFrame0.shape[0])
print('Full Data Set (Columns):', PersonFrame0.shape[1])
print('#####')
print('Horizontal Data Set (Rows):', PersonFrame2.shape[0])
print('Horizontal Data Set (Columns):', PersonFrame2.shape[1])

```

```

print('Only Sam Data')
print(PersonFrame2.head())
print('#####')
#####

```

Output:

```

In [47]: runfile('C:/VKHCG/01-Vermeulen/05-Organise/Organize-Secure-
Vault.py', wdir='C:/VKHCG/01-Vermeulen/05-Organise')

```

```

#####
Working Base : C:/VKHCG using win32
#####
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI

#####
Storing : C:/VKHCG/99-DW/datamart.db
Table: Dim-BMI-Secure

#####
#####
Loading : C:/VKHCG/99-DW/datamart.db Table: Dim-BMI-Secure
#####
#####
Full Data Set (Rows): 1080
Full Data Set (Columns): 5
#####
Horizontal Data Set (Rows): 692
Horizontal Data Set (Columns): 4
Only Sam Data
  Indicator  Height  Weight Name
0          4      1.0      35  Sam
1          4      1.0      40  Sam
2          4      1.0      45  Sam
3          4      1.0      50  Sam
4          4      1.0      55  Sam
#####

```

Practical 9

Reporting Data

A. Create a Network Routing Diagram

```
#####
import sys
import os
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
#####
pd.options.mode.chained_assignment = None
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
#####
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
sInputFileName='02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv'
#####
sOutputFileName1='05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company.gml'
sOutputFileName2='05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company.png'
Company='01-Vermeulen'
#####
#####
### Import Country Data
#####
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#####')
print('Loading :',sFileName)
print('#####')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('#####')
#####
print(CompanyData.head())
print(CompanyData.shape)
```

```
#####
G=nx.Graph()
for i in range(CompanyData.shape[0]):
    for j in range(CompanyData.shape[0]):
        Node0=CompanyData['Company_Country_Name'][i]
        Node1=CompanyData['Company_Country_Name'][j]
        if Node0 != Node1:
            G.add_edge(Node0,Node1)

for i in range(CompanyData.shape[0]):
    Node0=CompanyData['Company_Country_Name'][i]
    Node1=CompanyData['Company_Place_Name'][i] + '('+
CompanyData['Company_Country_Name'][i] + ')'
    if Node0 != Node1:
        G.add_edge(Node0,Node1)

print('Nodes:', G.number_of_nodes())
print('Edges:', G.number_of_edges())
#####
sFileName=Base + '/' + Company + '/' + sOutputFileName1
print('#####')
print('Storing :',sFileName)
print('#####')
nx.write_gml(G, sFileName)
#####
sFileName=Base + '/' + Company + '/' + sOutputFileName2
print('#####')
print('Storing Graph Image:',sFileName)
print('#####')
plt.figure(figsize=(15, 15))
pos=nx.spectral_layout(G,dim=2)
nx.draw_networkx_nodes(G,pos, node_color='k', node_size=10, alpha=0.8)
nx.draw_networkx_edges(G, pos,edge_color='r', arrows=False, style='dashed')
nx.draw_networkx_labels(G,pos,font_size=12,font_family='sans-serif',font_color='b')
plt.axis('off')
plt.savefig(sFileName,dpi=600)
plt.show()
#####
print('#####')
print('### Done!! #####')
```

```
print('#####')
#####
```

Output:

```
In [63]: runfile('C:/VKHCG/01-Vermeulen/05-Organise/Organise-Network-Routing-Company.py', wdir='C:/VKHCG/01-Vermeulen/05-Organise')
```

```
#####
Working Base : C:/VKHCG using win32
#####
Loading : C:/VKHCG/01-Vermeulen/02-Assess/01-EDS/02-Python/Assess-Network-Routing-Company.csv
#####
#####
Company_Country_Code ... Company_Country_Name
0 US ... United States of America
1 US ... United States of America
2 US ... United States of America
3 US ... United States of America
4 US ... United States of America

[5 rows x 5 columns]
(150, 5)
Nodes: 6
Edges: 6
#####
Storing : C:/VKHCG/01-Vermeulen/05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company.gml
#####
Storing Graph Image: C:/VKHCG/01-Vermeulen/05-Organise/01-EDS/02-Python/Organise-Network-Routing-Company.png
#####
```


Practical 9B

Directed Acyclic Graph

```
#####
import networkx as nx
import matplotlib.pyplot as plt
import sys
import os
import pandas as pd
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + 'VKHCG'
else:
    Base='C:/VKHCG'
#####
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
sInputFileName='01-Retrieve/01-EDS/02-Python/Retrieve_Router_Location.csv'
sOutputFileName1='Assess-DAG-Company-Country.png'
sOutputFileName2='Assess-DAG-Company-Country-Place.png'
Company='01-Vermeulen'
#####
### Import Company Data
#####
sFileName=Base + '/' + Company + '/' + sInputFileName
print('#####')
print('Loading :',sFileName)
print('#####')
CompanyData=pd.read_csv(sFileName,header=0,low_memory=False, encoding="latin-1")
print('Loaded Company :',CompanyData.columns.values)
print('#####')
#####
print(CompanyData)
print('#####')
print('Rows : ',CompanyData.shape[0])
print('#####')
#####
G1=nx.DiGraph()
G2=nx.DiGraph()
```

```
#####
for i in range(CompanyData.shape[0]):
    G1.add_node(CompanyData['Country'][i])
    sPlaceName= CompanyData['Place_Name'][i] + '-' + CompanyData['Country'][i]
    G2.add_node(sPlaceName)

print('#####')
for n1 in G1.nodes():
    for n2 in G1.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G1.add_edge(n1,n2)
print('#####')
print("Nodes of graph: ")
print(G1.nodes())
print("Edges of graph: ")
print(G1.edges())
print('#####')
#####
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####
sFileName=sFileDir + '/' + sOutputFileName1
print('#####')
print('Storing :', sFileName)
print('#####')
nx.draw(G1,pos=nx.spectral_layout(G1),
        nodecolor='r',edge_color='g',
        with_labels=True,node_size=8000,
        font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display
#####
print('#####')
for n1 in G2.nodes():
    for n2 in G2.nodes():
        if n1 != n2:
            print('Link :',n1,' to ', n2)
            G2.add_edge(n1,n2)
```

```

print('#####')
print("Nodes of graph: ")
print(G2.nodes())
print("Edges of graph: ")
print(G2.edges())
print('#####')
sFileDir=Base + '/' + Company + '/02-Assess/01-EDS/02-Python'
if not os.path.exists(sFileDir):
    os.makedirs(sFileDir)
#####
sFileName=sFileDir + '/' + sOutputFileName2
print('#####')
print('Storing :', sFileName)
print('#####')
nx.draw(G2,pos=nx.spectral_layout(G2),
        nodecolor='r',edge_color='b',
        with_labels=True,node_size=8000,
        font_size=12)
plt.savefig(sFileName) # save as png
plt.show() # display

```

Output:

```

In [16]: runfile('C:/VKHCG/01-Vermeulen/02-Assess/Assess-DAG-
Location.py', wdir='C:/VKHCG/01-Vermeulen/02-Assess')
#####
Working Base : C:/VKHCG using win32
#####
Loading : C:/VKHCG/01-Vermeulen/01-Retrieve/01-EDS/02-Python/
Retrieve_Router_Location.csv
#####
Loaded Company : ['Country' 'Place_Name' 'Latitude' 'Longitude']
#####

```

	Country	Place_Name	Latitude	Longitude
0	US	New York	40.7528	-73.9725
1	US	New York	40.7214	-74.0052
2	US	New York	40.7662	-73.9862
3	US	New York	40.7449	-73.9782
4	US	New York	40.7605	-73.9933
5	US	New York	40.7588	-73.9680
6	US	New York	40.7637	-73.9727
7	US	New York	40.7553	-73.9924
8	US	New York	40.7308	-73.9975
9	US	New York	40.7694	-73.9609
10	US	New York	40.7330	-74.0078
11	US	New York	40.7505	-73.9931
12	US	New York	40.7517	-73.9972
13	US	New York	40.7082	-74.0132
14	US	New York	40.7267	-73.9981
15	US	New York	40.7168	-73.9861
16	US	New York	40.7317	-73.9885
17	US	New York	40.7584	-73.9794
18	US	New York	40.7592	-73.9778
19	US	New York	40.7055	-74.0050
20	US	New York	40.6888	-74.0203
21	US	New York	40.7089	-74.0012
22	US	New York	40.7391	-73.9826

Practical 9 C

Graphics

CODE

```

import sys
import os
import pandas as pd
import matplotlib as ml
from matplotlib import pyplot as plt
#####
if sys.platform == 'linux':
    Base=os.path.expanduser('~') + '/VKHCG'
else:
    Base='C:/VKHCG'
print('#####')
print('Working Base :',Base, ' using ', sys.platform)
print('#####')
#####
GBase = Base+'/01-Vermeulen/06-Report/01-EDS/02-Python/'
ml.style.use('ggplot')

data=[
['London',      29.2, 17.4],
['Glasgow',     18.8, 11.3],
['Cape Town',   15.3, 9.0],
['Houston',     22.0, 7.8],
['Perth', 18.0, 23.7],
['San Francisco', 11.4, 33.3]
]
os_new=pd.DataFrame(data)
pd.Index(['Item', 'Value', 'Value Percent', 'Conversions', 'Conversion Percent',
        'URL', 'Stats URL'],
        dtype='object')

os_new.rename(columns = {0 : "Warehouse Location"}, inplace=True)
os_new.rename(columns = {1 : "Profit 2016"}, inplace=True)
os_new.rename(columns = {2 : "Profit 2017"}, inplace=True)

explode = (0, 0, 0, 0, 0, 0.1)
labels=os_new['Warehouse Location']

```

