



Converting ACL to python

Virtual Community Event
August 21, 2024

ABOUT ✓AUDITOPiA



Online Community

An online community for audit/GRC professionals to collaborate and add value to each of our organizations.



Knowledge Bank

A knowledge bank of resources and tools where we can contribute and comment.



Learning Platform

A learning platform to tap into our collective knowledge in live events and self-paced courses.

HERE IS HOW YOU CAN GET INVOLVED

If you would like to help us build the Auditopia community, email us at info@auditopia.com!



Engage

Visit [Auditopia.com](https://auditopia.com) often to check out new content, ask questions, share your thoughts, and give us feedback!



Facilitate

Become a [collaboration group organizer](#), curating content in technical or non-technical subjects



Teach

Share from your [knowledge and skills](#), through live events, courses, blog posting, resources, and more



Coordinate

Help us [schedule](#) events, [organize](#) content, and ensure our content creators have everything they need to be successful



Develop

Work on technical [projects](#) and provide [troubleshooting](#) help on the website, during [events](#), and other areas

WE'D LOVE FOR YOU TO JOIN IN TODAY!



Chat

Share your thoughts on today's topics. Let us know where you are joining from!



Q&A

Feel free to use the Q&A function in Zoom at any time to ask questions.



Polls

We encourage you to participate, for a more engaging experience.



Survey

Please complete our end of session survey. *Feedback makes us better!*

Actively engage to share your thoughts and insights with the community. Your input will make the session more dynamic and valuable for everyone!



Auditopia Team Member

Managing Director
Aufinia
Claire WORLEDGE
cjw@aufinia.com

I'm your host today TODAY!



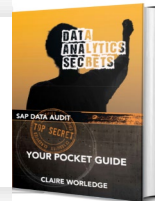
10 years at Deloitte

13 years running Aufinia



CISA, CFE, ACL Certified Data Analyst, SAP data expert

Author: Data Analytics Secrets: SAP data audit:
YOUR POCKET GUIDE



Likes to run marathons

Interested in bio-hacking

Lives in Danang



Our Session **Today**



Ice Breaker



How to get into Python in 5 minutes
Free guide!



Convert ACL commands to Python
20 examples! (+1 extra bonus!)



Naming convention for Python
Accelerate your Python development



Wrap Up



ICE BREAKER



Ice Breaker

Auditopia Community Meet & Greet

Where are you from?



*Please share
your answer in
the chat with
the community!*

*(Make sure to select
"To: Everyone")*



Poll 1: Do you already use Python in your organization?

We'll be using the **polling feature**. Select your answer to these questions. *Votes are anonymous!*

Do you already use Python in your organization?





Poll 2: Do you plan to use Python in your organization

We'll be using the **polling feature**. Select your answer to these questions. *Votes are anonymous!*

Do you plan to use Python in your organization?

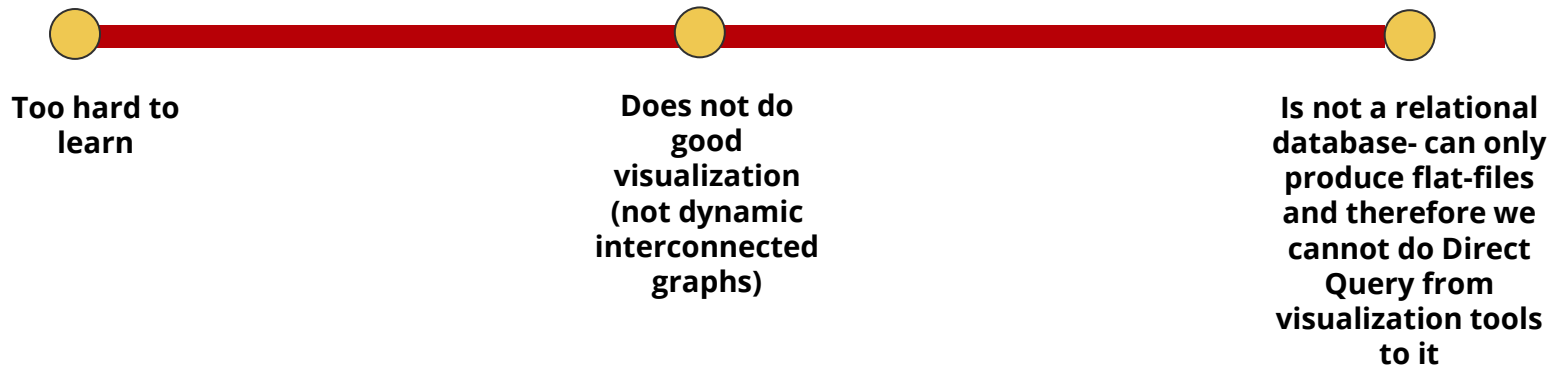




Poll 3: What inconveniences do you see for using something like Python?

We'll be using the **polling feature**. Select your answer to these questions. *Votes are anonymous!*

What inconveniences do you see for using something like Python?



Any other inconveniences – let us know in the chat!



How to get into Python in 5 minutes



How to get into Python in 5 minutes

Get your free guide!

- You will need to download and install the following free items (links mentioned in the document):
 1. Visual Studio Code (easiest to code editor to use)
 2. Python
- You can also download Python “libraries” which enable you to do lots of different things, from Web-Scrapping, to Predictive Analysis to Reading and categorizing paragraphs of text



How To Get Into
Python In 5 Minute

A blurred background image showing a group of people sitting around a table, working on laptops. The scene is brightly lit, possibly from a window. The text 'ACL -> Python' is overlaid in the center.

ACL -> Python



ACL to Python

FREE copy of all the Python scripts analyzed here!

For a FREE copy of the scripts displayed on the following pages, please go here:

[https://drive.google.com/file/d/1WIAAYg9iIX_c_xLiklu2q3gAnfYfOlAw/view?usp=drive link](https://drive.google.com/file/d/1WIAAYg9iIX_c_xLiklu2q3gAnfYfOlAw/view?usp=drive_link)



ACL command: IMPORT

Creates an Analytics table by defining and importing a delimited text file.

```
# Step 1/ Import the library
import pandas as PI_PANDAS

# Step 2/ Set the variables that will be used in the script
ZV_SOURCE_FOLDER = r'C:\CLAIRE_BACKUP\2021_AUFINIA_WEBINARS\148_PYTHON_FOR_AUDITORS\01_PYTHON_SCRIPTS\A_PHASE'

# Step 3/ Import the supplier movements (closed items)
D01_01_RT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')

# Step 4/ Print the DataFrame
print(D01_01_RT_DF_BSAK)
```

Explanation of python script

- Import the pandas library, that helps us to be able to read and write to files
- Use the read_csv function of the pandas library



ACL command: EXPORT

Exports data from Analytics to the specified file format, or to Results in HighBond.

```
# Step 1/ Import the library
```

```
import pandas as PI_PANDAS
```

```
# Step 2/ Set the variables that will be used in the script
```

```
ZV_SOURCE_FOLDER = r'C:\CLAIRE_BACKUP\2021_AUFINIA_WEBINARS\148_PYTHON_FOR_AUDITORS\01_PYTHON_SCRIPTS\A_PHASE'
```

```
ZV_RESULTS_FOLDER = r'C:\CLAIRE_BACKUP\2021_AUFINIA_WEBINARS\148_PYTHON_FOR_AUDITORS\01_PYTHON_SCRIPTS\STORE_DATA'
```

```
# Step 3/ Import the supplier movements (closed items)
```

```
D01_01_RT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')
```

```
# Step 4/ Export the DataFrame to csv file and print it
```

```
D01_01_RT_DF_BSAK.to_csv(ZV_RESULTS_FOLDER + '\D01_01_RT_DF_BSAK.csv', sep='\t')
```

```
print(D01_01_RT_DF_BSAK)
```

Explanation of python script

- Use the `.to_csv` function of the pandas library and export to a tab delimited file



ACL command: DEFINE FIELD

Defines a physical data field in an Analytics table layout.

```
# Step 3/ Import the supplier movements (closed items)
D03_01_TT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')

# Step 4/ Create the signed (positive/negative) amount in local currency
D03_01_TT_DF_BSAK['ZF_BSAK_DMBTR_SIGNED'] = D03_01_TT_DF_BSAK.apply(lambda row: -row['DMBTR'] if row['SHKZG'] == 'H' else row['DMBTR'], axis=1)

# Step 5/ Select the key fields for accounting document line item and signed value
D03_02_RT_DF_BSAK_SELECTED_COLUMN = D03_01_TT_DF_BSAK[['BUKRS', 'GJAHR', 'BELNR', 'BUZEI', 'ZF_BSAK_DMBTR_SIGNED']]

# Step 6/ Export the DataFrame to a CSV file and print it
D03_02_RT_DF_BSAK_SELECTED_COLUMN.to_csv(ZV_RESULTS_FOLDER + '\D03_02_RT_DF_BSAK_SELECTED_COLUMN.csv', sep='\t', index=False)
print(D03_02_RT_DF_BSAK_SELECTED_COLUMN)
```

Explanation of python script

- Create the field ZF_BSAK_DMBTR_SIGNED. It fills this column with the negative value of column 'DMBTR' if the value in column 'SHKZG' is 'H', otherwise it leaves the value of column 'DMBTR' unchanged.



ACL command: EXTRACT

Extracts data from an Analytics table and outputs it to a new Analytics table, or appends it to an existing Analytics table. You can extract entire records or selected fields.

```
# Step 1/ Import the library
import pandas as PI_PANDAS

# Step 2/ Set the variables that will be used in the script
ZV_SOURCE_FOLDER = r'C:\CLAIRE_BACKUP\2021_AUFINIA_WEBINARS\148_PYTHON_FOR_AUDITORS\01_PYTHON_SCRIPTS\A_PHASE '

# Step 3/ Import the DataFrame
D04_01_TT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')

# Step 4/ Create a copy of the 'BSAK' Dataframe
D04_02_RT_DF_BASK_COPIED = D04_01_TT_DF_BSAK.copy()

# Step 5/ Print the dataframe
print(D04_02_RT_DF_BASK_COPIED)
```

Explanation of python script

- Create a copy of a dataframe



ACL command: SORT

Sorts records in an Analytics table into an ascending or descending sequential order, based on a specified key field or fields. The results are output to a new, physically reordered Analytics table.

```
# Step 3/ Import the closed supplier movements
D05_01_TT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')

# Step 4/ Sort ascending date for each supplier
D05_02_RT_DF_BSAK_SORT_ASCENDING = D05_01_TT_DF_BSAK.sort_values(by=['LIFNR', 'BUDAT'], ascending=[True, True])
print("Sorted by SUPPLIER and DATE (Ascending):\n", D05_02_RT_DF_BSAK_SORT_ASCENDING)

# Step 5/ Sort descending date for each supplier
D05_03_RT_DF_BSAK_SORT_DESCENDING = D05_01_TT_DF_BSAK.sort_values(by=['LIFNR', 'BUDAT'], ascending=[True, False])
print("Sorted by SUPPLIER and DATE (Descending):\n", D05_03_RT_DF_BSAK_SORT_DESCENDING)
```

Explanation of python script

- Sort the data frame in ascending or descending order



ACL command: CLASSIFY/ SUMMARIZE

Groups records based on identical values in a character or numeric field. Counts the number of records in each group, and also subtotals specified numeric fields for each group.

```
# Step 4/ Calculate the signed (+/-) amount in local currency
D06_03_RT_DF_BSAK_BSIK_CONCATED['ZF_BSAK_DMBTR_SIGNED'] = D06_03_RT_DF_BSAK_BSIK_CONCATED.apply(lambda row: -row['DMBTR']
if row['SHKZG'] == 'H' else row['DMBTR'], axis=1)

# Step 5/ Calculate the total per vendor
D06_04_RT_DF_BSAK_LIFNR_DMBTR_TOT = D06_03_RT_DF_BSAK_BSIK_CONCATED.groupby('LIFNR')['ZF_BSAK_DMBTR_SIGNED'].sum().
reset_index()

# Step 6/ Calculate the total per vendor, fiscal year
D06_05_RT_DF_BSAK_LIFNR_GJAHR_DMBTR_TOT = D06_03_RT_DF_BSAK_BSIK_CONCATED.groupby(['LIFNR', 'GJAHR'])
['ZF_BSAK_DMBTR_SIGNED'].sum().reset_index()

# Step 7/ Set display option to two decimal places
PI_PANDAS.set_option('display.float_format', lambda x: '%.2f' % x)

# Step 8/ Print the result
print("Total per vendor:")
print(D06_04_RT_DF_BSAK_LIFNR_DMBTR_TOT)

print("\nTotal per vendor per year:")
print(D06_05_RT_DF_BSAK_LIFNR_GJAHR_DMBTR_TOT)
```

Explanation of python script

- Calculate the total value per vendor.
- Calculate the total value per vendor, year.



ACL command: JOIN

Combines fields from two Analytics tables into a new, single Analytics table.

```
# Step 3/ Import the general ledger detail and header
D07_01_TT_DF_BSEG = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSEG.csv', sep='\t')
D07_02_TT_DF_BKPF = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BKPF.csv', sep='\t')

# Step 4/ Filter on accounting documents that are not pre-posting
D07_02_TT_DF_BKPF['BSTAT'].fillna('', inplace=True)
D07_03_TT_DF_BSEG_BKPF_FILTERED = D07_02_TT_DF_BKPF[D07_02_TT_DF_BKPF['BSTAT'] == '']

# Step 4/ Add the header information to the detailed journal entry lines
D07_04_RT_DF_BSEG_BKPF_MERGED = PI_PANDAS.merge(D07_01_TT_DF_BSEG, D07_03_TT_DF_BSEG_BKPF_FILTERED, on=['BUKRS', 'GJAHR', 'BELNR'], how='inner')

# Step 5/ Print the result
print(D07_04_RT_DF_BSEG_BKPF_MERGED)
```

Explanation of python script

- Join the journal entry header information to the journal entry detail table.



ACL command: APPEND

Combines records from two or more Analytics tables by appending them in a new Analytics table.

```
# Step 3/ Import the supplier movements
D08_01_TT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')
D08_02_TT_DF_BSIK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSIK.csv', sep='\t')

# Step 4/ Append the open and closed items together
D08_03_RT_DF_BSAK_BSIK_CONCATED = PI_PANDAS.concat([D08_01_TT_DF_BSAK, D08_02_TT_DF_BSIK])

# Step 5/ Print the result
print(D08_03_RT_DF_BSAK_BSIK_CONCATED)
```

Explanation of python script

- Concatenate the supplier closed movements with the supplier open items (similar to a UNION in SQL)



ACL command: CROSSTAB

Groups records based on identical combinations of values in two or more character or numeric fields, and displays the resulting groups in a grid of rows and columns. Counts the number of records in each group, and also subtotals specified numeric fields for each group.

```
# Step 3/ Import the general ledger detail
D09_01_TT_DF_BSEG = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSEG.csv', sep='\t')

# Step 4/ Filter on debit lines because later we will sum and debit = credit
D09_02_TT_DF_BSEG_FILTERED = D09_01_TT_DF_BSEG[D09_01_TT_DF_BSEG['SHKZG'] == 'S']

# Step 5/ Create a pivot table to show total journal entry value per company code, fiscal year
D09_03_RT_DF_BSEG_PIVOTED = D09_02_TT_DF_BSEG_FILTERED.pivot_table(index='BUKRS', columns='GJAHR', values='DMBTR', aggfunc='sum', fill_value=0)

# Step 6/ Set display option to two decimal places
PI_PANDAS.set_option('display.float_format', lambda x: '%.2f' % x)

# Step 7/ Save the pivot table to a CSV file and print the pivot table
D09_03_RT_DF_BSEG_PIVOTED.to_csv(ZV_RESULTS_FOLDER + '\D09_03_RT_DF_BSEG_PIVOTED.csv', sep='\t')
print(D09_03_RT_DF_BSEG_PIVOTED)
```

Explanation of python script

- Create a pivot table with:
 - Company code as row
 - Fiscal year as column
 - Value summed
- Adjust the number format to 2 decimal places



ACL command: COUNT

Counts the total number of records in the current view.

```
# Step 3/ Import the supplier closed movements
D10_01_TT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')

# Step 4/ Get the average value of the value in local currency
ZV_AVG_DMBTR = D10_01_TT_DF_BSAK['DMBTR'].mean()

# Step 5/ Count the number of lines that have a value that is greater than the average
ZV_COUNT_VAL_GT_AVG = D10_01_TT_DF_BSAK[D10_01_TT_DF_BSAK['DMBTR'] > ZV_AVG_DMBTR]['DMBTR'].count()

# Step 6/ Display the number of lines that are greater than the average
print('The number of values greater than the average is: ', ZV_COUNT_VAL_GT_AVG)
```

Explanation of python script

- Count the number of supplier movements that are above the average, in terms of value.



ACL command: STATISTICS

Calculates statistics for one or more numeric or datetime fields in an Analytics table.

```
# Step 3/ Import the supplier closed movements
D11_01_TT_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_BSAK.csv', sep='\t')

# Step 4/ Generate descriptive statistics for numeric columns
D11_02_RT_DF_BSAK_NUMERIC_STATS= D11_01_TT_DF_BSAK.describe()

# Step 5/ Set display format to two decimal places
PI_PANDAS.set_option('display.float_format', lambda x: '%.2f' % x)

# Step 6/ Print the summary statistics
print("Summary statistics for numeric columns:")
print(D11_02_RT_DF_BSAK_NUMERIC_STATS)
```

Explanation of python script

- Do statistics on all numeric fields in the supplier movements table



ACL command: VERIFY

Checks for data validity errors in one or more fields in an Analytics table by verifying that the data is consistent with the field definitions in the table layout.

```
# Step 3/ Import the currency factors
D12_01_TT_DF_TCURX = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '\A_TCURX.csv', sep='\t')

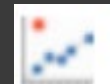
# Step 4/ Look for NULL or non-integer values in each column of the table

ZV_DI_ERRORS = {}
for ZV_COLUMN in D12_01_TT_DF_TCURX.columns:
    if D12_01_TT_DF_TCURX[ZV_COLUMN].isna().any():
        ZV_DI_ERRORS[ZV_COLUMN] = f"NaN (missing values) found in column {ZV_COLUMN}"
    try:
        D12_01_TT_DF_TCURX[ZV_COLUMN] = D12_01_TT_DF_TCURX[ZV_COLUMN].astype(int)
    except ValueError:
        ZV_DI_ERRORS[ZV_COLUMN] = f"Non-integer data found in column {ZV_COLUMN}"

# Step 5/ Iterate through the dictionary of error messages and print each message
for ZV_COLUMN, ZV_ERROR_MESSAGE in ZV_DI_ERRORS.items():
    print(ZV_ERROR_MESSAGE)
```

Explanation of python script

- Check if there are any non-integer or missing values (NULL) in the currency decimals table,



ACL command: OUTLIERS

Identifies statistical outliers in a numeric field. Outliers can be identified for the field as a whole, or for separate groups based on identical values in one or more character, numeric, or datetime key fields.

```
# Step 5/ Create an array that will hold the list of accounts that have values that are outside
#         standard deviation
ZV_AR_HIGH_DEVIATION_ACCOUNT = []

# Step 6/ Create a list of general ledger accounts for which the standard deviation is greater than the mean
for ZV_BSEG_HKONT in D13_02_TT_DF_BSEG_FILTER['HKONT'].unique():
    # 6.1/ Filter on the current general ledger account and get the column for value in local currency
    D13_03_TT_DF_BSEG_FILTER = D13_02_TT_DF_BSEG_FILTER[D13_02_TT_DF_BSEG_FILTER['HKONT'] == ZV_BSEG_HKONT][['DMBTR']]
    # 6.2/ Calculate the mean
    ZV_MEAN_VAL = D13_03_TT_DF_BSEG_FILTER.mean()
    # 6.3/ Calculate the standard deviation
    ZV_STD_VAL = D13_03_TT_DF_BSEG_FILTER.std()
    # 6.4/ If the standard deviation is greater than the mean add the account to the list
    if ZV_STD_VAL > ZV_MEAN_VAL:
        ZV_AR_HIGH_DEVIATION_ACCOUNT.append(ZV_BSEG_HKONT)

# Step 7/ Apply filter based on the toggle

if ZV_APPLY_DEVIATION_FILTER:
    # 7.1/ Filter DataFrame to keep only rows where 'HKONT' column is in ZV_AR_HIGH_DEVIATION_ACCOUNT
    D13_04_TT_DF_BSEG_FILTER = D13_02_TT_DF_BSEG_FILTER[D13_02_TT_DF_BSEG_FILTER['HKONT'].isin(ZV_AR_HIGH_DEVIATION_ACCOUNT)]
    print("Filtering enabled, accounts filtered:", len(ZV_AR_HIGH_DEVIATION_ACCOUNT))
    print(f"Data points after filter: {len(D13_04_TT_DF_BSEG_FILTER)}") # Number of data points after filter
else:
    # 7.2/ Reset if not filtering
    D13_04_TT_DF_BSEG_FILTER = D13_01_TT_DF_BSEG.copy()
    print("No filtering applied.")
    print(f"Total data points without filter: {len(D13_04_TT_DF_BSEG_FILTER)}") # Total data points without filter
    print(f"Unique HKONT without filter: {len(D13_04_TT_DF_BSEG_FILTER['HKONT'].unique())}") # Unique accounts without filter
```

Explanation of python script

- Calculate the mean and standard deviation per general ledger account in the general ledger.
- Flag accounts for which unusual values (outliers) are found.
- Create a graph to highlight the journal entries with unusual items.



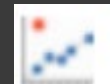
```
# Step 8/ Create a plot to show one dot per document number for each general ledger account
ZV_OB_FIGURE = PI_PLOTLY_GRAPH_OBJECTS.Figure()

for ZV_BSEG_HKONT in D13_04_TT_DF_BSEG_FILTER['HKONT'].unique():
    D13_05_XT_DF_BSEG_FILTER = D13_04_TT_DF_BSEG_FILTER[D13_04_TT_DF_BSEG_FILTER['HKONT'] == ZV_BSEG_HKONT]['DMBTR']
    ZV_OB_FIGURE.add_trace(PI_PLOTLY_GRAPH_OBJECTS.Box(y=D13_05_XT_DF_BSEG_FILTER, name=str(ZV_BSEG_HKONT), boxpoints='outliers'))
ZV_OB_FIGURE.update_layout(title='Distribution of value per account', yaxis_title='Value in local currency', xaxis_title='GL account')

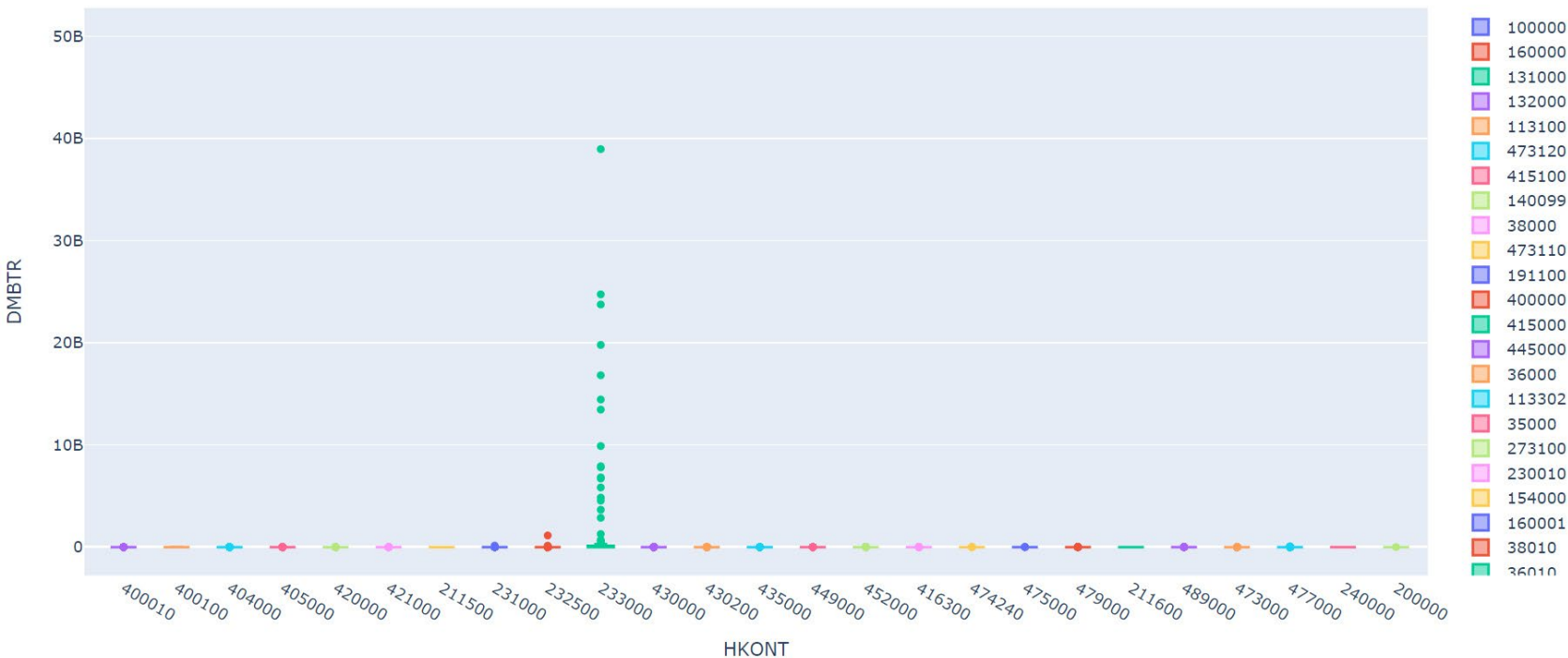
# Step 9/ Save the graph as a HTML file
ZV_OB_FIGURE.write_html("DistributionOfValuePerAccount.html")
```

Explanation of python script

- Show values that are outside of the standard deviation and make a dynamic plot that can be seen in a Web browser



Distribution of DMBTR per HKONT





ACL command: SEQUENCE

Determines if one or more fields in an Analytics table are in sequential order, and identifies out-of-sequence items.

```
# Step 7/ Create concatenated field for checking
D14_06_TT_DF_BSEG_BKPF_SORTED['ZF_BUKRSGJAHRBELNRBUZEI'] = D14_06_TT_DF_BSEG_BKPF_SORTED['BUKRS'].astype(str) +
D14_06_TT_DF_BSEG_BKPF_SORTED['GJAHR'].astype(str) + D14_06_TT_DF_BSEG_BKPF_SORTED['BELNR'].astype(str) +
D14_06_TT_DF_BSEG_BKPF_SORTED['BUZEI'].astype(str)
# Create a shifted version of ZF_BUKRSGJAHRBELNRBUZEI
D14_06_TT_DF_BSEG_BKPF_SORTED['ZF_BUKRSGJAHRBELNRBUZEI_SHIFT'] = D14_06_TT_DF_BSEG_BKPF_SORTED['ZF_BUKRSGJAHRBELNRBUZEI'].shift(1)
# Checking if each concatenated field is greater than the previous one
D14_06_TT_DF_BSEG_BKPF_SORTED['ZF_BUKRSGJAHRBELNRBUZEI_CHKSEQ'] = D14_06_TT_DF_BSEG_BKPF_SORTED['ZF_BUKRSGJAHRBELNRBUZEI'].shift(1)
<= D14_06_TT_DF_BSEG_BKPF_SORTED['ZF_BUKRSGJAHRBELNRBUZEI']

# Step 8/ Find where the sequence breaks
D14_07_XT_DF_BSEG_BKPF = D14_06_TT_DF_BSEG_BKPF_SORTED[D14_06_TT_DF_BSEG_BKPF_SORTED['ZF_BUKRSGJAHRBELNRBUZEI_CHKSEQ'] == False]

# Step 9/ Display results
print("Sequence Errors: (if any)")
if not D14_07_XT_DF_BSEG_BKPF.empty:
    # Include the new fields in the output for better analysis
    print(D14_07_XT_DF_BSEG_BKPF[['ZF_BUKRSGJAHRBELNRBUZEI', 'ZF_BUKRSGJAHRBELNRBUZEI_SHIFT']])
else:
    print("No sequence errors found.")
```

Explanation of python script

- Compares values on different rows using shift()
- If the following value is lower than the previous, then filter
- Display the information about the error.



ACL command: DUPLICATES

Detects whether duplicate values or entire duplicate records exist in an Analytics table.

```
# Step 5/ Identify how many rows there are per combination of external document number, document date, vendor and document amount
D15_03_TT_DF_BSAK_BSIK_INV_KEY_COUNT = D15_02_TT_DF_BSAK_BSIK_FILTERED.groupby(['XBLNR', 'BLDAT', 'LIFNR',
'WRBTR']).size().reset_index(name='ZF_COUNT_LINES_PER_DUP_KEY')

# Step 6/ Filter on lines that are duplicates on document number, date, document amount, vendor
D15_04_TT_DF_BSAK_BSIK_INV_KEYCOUNT_GT_1 = D15_03_TT_DF_BSAK_BSIK_INV_KEY_COUNT[D15_03_TT_DF_BSAK_BSIK_INV_KEY_COUNT
['ZF_COUNT_LINES_PER_DUP_KEY'] > 1]

# Step 6/ Obtain the detail (all lines for each case) from the original table
D15_05_TT_DF_BSAK_BSIK_DUPINV = PI_PANDAS.merge(D15_04_TT_DF_BSAK_BSIK_INV_KEYCOUNT_GT_1, D15_02_TT_DF_BSAK_BSIK_FILTERED,
how='inner',
on=['XBLNR', 'BLDAT', 'LIFNR', 'WRBTR'])

# Step 7/ Choose the columns for the output
D15_06_XT_DF_BSAK_BSIK_DUPINV = D15_05_TT_DF_BSAK_BSIK_DUPINV[['BUKRS','GJAHR','BELNR','BUZEI','LIFNR','XBLNR','BLDAT','WRBTR']]

# Step 8/ View and export the result
print(D15_06_XT_DF_BSAK_BSIK_DUPINV)
D15_06_XT_DF_BSAK_BSIK_DUPINV.to_csv(ZV_RESULTS_FOLDER + '\D15_06_XT_DF_BSAK_BSIK_DUPINV.csv', sep='\t', index=False)
```

Explanation of python script

- Use size() to know the number of records per combination and then filter on those that are > 1 (these are the duplicate cases).



ACL command: GAPS

Detects whether a numeric or datetime field in an Analytics table contains one or more gaps in sequential data.

```
# Step 5/ Create a list of company codes and fiscal year
# with a column BELNR that has the entire lists of BELNR
D16_03_XT_DF_BSEG_GROUPED = D16_02_TT_DF_BSEG_FILTERED.groupby(['BUKRS', 'GJAHR'])['BELNR'].apply(list).reset_index()

# Step 6/ Check for gaps
def FC_BSEG_CHECK_GAPS(ZVFCI_INTEGER):
    ZV_AR_BELNRS = sorted(ZVFCI_INTEGER['BELNR']) # Sort the list of BELNRS
    ZV_AR_GAPS = []
    for i in range(len(ZV_AR_BELNRS) - 1):
        if ZV_AR_BELNRS[i+1] - ZV_AR_BELNRS[i] > 1:
            ZV_AR_GAPS.append(f"Gap between BELNR number {ZV_AR_BELNRS[i]} and {ZV_AR_BELNRS[i+1]}")
    return ZV_AR_GAPS

# Step 7/ Apply the function to each group
# axis=1 means do the function row by row
D16_03_XT_DF_BSEG_GROUPED['ZF_BSEG_BELNR_GAPS'] = D16_03_XT_DF_BSEG_GROUPED.apply(FC_BSEG_CHECK_GAPS, axis=1)

# Step 8/ Print results
print(D16_03_XT_DF_BSEG_GROUPED[['BUKRS', 'GJAHR', 'ZF_BSEG_BELNR_GAPS']])
```

Explanation of python script

- Compare the document numbers between row [i] and [i+1] to see if there are any missing document numbers between them



ACL command: STRATIFY

Groups records into numeric intervals based on values in a numeric field. Counts the number of records in each interval, and also subtotals specified numeric fields for each interval.

```
# Step 6/ Define the intervals based on the maximum DMBTR, ensure the last interval includes ZV_DMBTR_MAX
ZV_AR_INTERVALS = [(i * ZV_DMBTR_MAX / 10, (i + 1) * ZV_DMBTR_MAX / 10) if i < 9 else (i * ZV_DMBTR_MAX / 10, ZV_DMBTR_MAX + 1) for
i in range(10)]

# Step 7/ Define a function to assign each DMBTR to its corresponding interval
def FC_ASSIGN_INTERVAL(ZVFCI_FLOAT):
    for i, (ZV_START, ZV_END) in enumerate(ZV_AR_INTERVALS):
        if ZV_START <= ZVFCI_FLOAT < ZV_END:
            return f'from ${ZV_START:.2f} to ${ZV_END:.2f}'

# Step 8/ Apply the function to create a new column for the intervals
D17_02_TT_DF_BSEG_FILTERED['ZF_BSEG_DMBTR_INTERVAL'] = D17_02_TT_DF_BSEG_FILTERED['DMBTR'].apply(FC_ASSIGN_INTERVAL)

# Step 9/ Group by the intervals and calculate the total value and number of lines per interval
D17_03_RT_DF_BSEG_GROUPED = D17_02_TT_DF_BSEG_FILTERED.groupby('ZF_BSEG_DMBTR_INTERVAL').agg(
    ZF_BSEG_DMBTR_SUM=PI_PANDAS.NamedAgg(column='DMBTR', aggfunc='sum'),
    ZF_BSEG_BUKRS_GJAHR_BELNR_BUZEI_COUNT=PI_PANDAS.NamedAgg(column='DMBTR', aggfunc='count')
).reset_index()

# Step 10/ Format the amount in local currency to two decimal places
D17_03_RT_DF_BSEG_GROUPED['ZF_BSEG_DMBTR_SUM'] = D17_03_RT_DF_BSEG_GROUPED['ZF_BSEG_DMBTR_SUM'].apply(lambda x: '%.2f' % x)

# Step 11/ Print the result and store in a tab-delimited file
print(D17_03_RT_DF_BSEG_GROUPED)
D17_03_RT_DF_BSEG_GROUPED.to_csv(ZV_RESULTS_FOLDER + '\D17_03_RT_DF_BSEG_GROUPED.csv', sep='\t', index=False)
```

Explanation of python script

- Create 10 intervals based on the maximum value in the general ledger
- Show the total value for each interval range



ACL command: AGEING

Groups records into aging periods based on values in a date or datetime field. Counts the number of records in each period, and also subtotals specified numeric fields for each period.

Explanation of python script

- Create ageing buckets and labels.
- Cut the data up into the ageing buckets and assign the labels

```
14 # Step 3/ Create labels and bins for ageing buckets
15 ZV_AR_BINS = [0, 30, 60, 90, 120, 150, 180, 365, float('inf')]
16 ZV_AR_LABELS = ['0-30', '31-60', '61-90', '91-120', '121-150', '151-180', '181-365', '>365']
17
18 # Step 4/ Import the supplier open and closed items
19 D18_01_DF_BSAK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '/A_BSAK.csv', sep='\t')
20 D18_02_DF_BSIK = PI_PANDAS.read_csv(ZV_SOURCE_FOLDER + '/A_BSIK.csv', sep='\t')
21
22 # Step 5/ Concatenate the supplier closed items with the supplier open items
23 D18_03_TT_DF_BSAK_BSIK_CONCATENATED = PI_PANDAS.concat([D18_01_DF_BSAK, D18_02_DF_BSIK])
24
25 # Step 6/ Filter on lines that have been posted
26 D18_03_TT_DF_BSAK_BSIK_CONCATENATED['BSTAT'].fillna('', inplace=True)
27 D18_04_TT_DF_BSAK_BSIK_FILTERED = D18_03_TT_DF_BSAK_BSIK_CONCATENATED[D18_03_TT_DF_BSAK_BSIK_CONCATENATED['BSTAT'] == '']
28
29 # Step 7/ - Ensure that the posting date is in date format
30 # - Calculate the number of days from posting date to cutoff date
31 # - Add an age range label based on the bins and labels arrays
32 # - Add the signed amount
33 D18_04_TT_DF_BSAK_BSIK_FILTERED['BUDAT'] = PI_PANDAS.to_datetime(D18_04_TT_DF_BSAK_BSIK_FILTERED['BUDAT'], format='%Y%m%d')
34 D18_04_TT_DF_BSAK_BSIK_FILTERED['ZF_BSAK_BUDAT_MINUS_CUTOFF'] = (ZV_CUT_OFF_DATE - D18_04_TT_DF_BSAK_BSIK_FILTERED['BUDAT']).dt.days
35 D18_04_TT_DF_BSAK_BSIK_FILTERED['ZF_BSAK_BUDAT_CUTOFF_RANGE'] = PI_PANDAS.cut(D18_04_TT_DF_BSAK_BSIK_FILTERED
36 ['ZF_BSAK_BUDAT_MINUS_CUTOFF'], bins=ZV_AR_BINS, labels=ZV_AR_LABELS, right=True)
37 D18_04_TT_DF_BSAK_BSIK_FILTERED['ZF_BSAK_DMBTR_SIGNED'] = D18_04_TT_DF_BSAK_BSIK_FILTERED.apply(lambda row: -row['DMBTR'] if row['SHKZG']
```



ACL command: SAMPLE

Draws a sample of records using either the record sampling or monetary unit sampling method.

```
# Step 4/ Set your cut off value
ZV_CUT_OFF = 1000

# Step 5/ Filter rows where the value in local currency is greater than the cut-off
D19_02_TT_DF_BSEG_FILTERED = D19_01_TT_DF_BSEG[D19_01_TT_DF_BSEG['DMBTR'] > ZV_CUT_OFF]

# Step 6/ Calculate a weight based on the value as a % of the total
ZV_BSEG_DMTBR_PERCTOT = D19_02_TT_DF_BSEG_FILTERED['DMBTR'] / D19_02_TT_DF_BSEG_FILTERED['DMBTR'].sum()

# Step 7/ Create a sample, using the weight
ZV_SAMPLE_SIZE = 25
D19_03_TT_DF_BSEG_SAMPLE = D19_02_TT_DF_BSEG_FILTERED.sample(n=ZV_SAMPLE_SIZE, weights=ZV_BSEG_DMTBR_PERCTOT)

# Step 8/ Select columns that are required in the output
D19_04_RT_DF_BSEG_SELECTED_COLUMN = D19_03_TT_DF_BSEG_SAMPLE[['BUKRS', 'GJAHR', 'BELNR', 'BUZEI', 'DMBTR']]

# Step 9/ Format the amount in local currency to two decimal places
D19_04_RT_DF_BSEG_SELECTED_COLUMN['DMBTR'] = D19_04_RT_DF_BSEG_SELECTED_COLUMN['DMBTR'].apply(lambda x: '%.2f' % x)

# Step 10/ Print the result
print(D19_04_RT_DF_BSEG_SELECTED_COLUMN)
```

Explanation of python script

- Allow for entry of a cut-off
- Calculate the weight based on the average DMBTR value
- Take a random sample for items > cut-off and considering the weight



ACL command: GROUP

Executes one or more ACLScript commands on a record before moving to the next record in the table, with only one pass through the table. Command execution can be controlled by conditions.

```
# Step 5/ Create a condition variable for filtering supplier movements
ZV_CONDITION = D20_01_TT_DF_BSAK['GJAHR'] == ZV_RUNTOT_START_YEAR

# Step 6/ Calculate the running total per vendor

if ZV_CONDITION.any():
    #5.1/ Filter on the condition
    D20_02_TT_DF_BSAK_FILTERED = D20_01_TT_DF_BSAK[ZV_CONDITION]
    #5.2/ Order by supplier, posting date
    D20_03_TT_DF_BSAK_SORTED = D20_02_TT_DF_BSAK_FILTERED.sort_values(by=['LIFNR', 'BUDAT'])
    #5.2/ Total value per supplier date
    D20_04_TT_DF_BSAK_GROUPED = D20_03_TT_DF_BSAK_SORTED.groupby(['LIFNR', 'BUDAT'])['ZF_BSAK_DMBTR_SIGNED'].sum().reset_index()
    D20_04_TT_DF_BSAK_GROUPED['ZF_BSAK_DMBTR_RUNTOT'] = D20_04_TT_DF_BSAK_GROUPED.groupby('LIFNR')['ZF_BSAK_DMBTR_SIGNED'].cumsum()
    D20_05_RT_DF_BSAK_LIFNR_DMBTRCUM = PI_PANDAS.DataFrame({
        'BSAK_LIFNR': D20_04_TT_DF_BSAK_GROUPED['LIFNR'],
        'BSAK_BUDAT': D20_04_TT_DF_BSAK_GROUPED['BUDAT'],
        'ZF_BSAK_DMBTR_SIGNED': D20_04_TT_DF_BSAK_GROUPED['ZF_BSAK_DMBTR_SIGNED'].apply(lambda x: '%.2f' % x),
        'ZF_BSAK_DMBTR_RUNTOT': D20_04_TT_DF_BSAK_GROUPED['ZF_BSAK_DMBTR_RUNTOT'].apply(lambda x: '%.2f' % x)
    })
    print(D20_05_RT_DF_BSAK_LIFNR_DMBTRCUM)
else:
    print("No supplier movements were found for fiscal year:", ZV_RUNTOT_START_YEAR)
```

Explanation of python script

- Calculate the running total per vendor, ordered per date
- Using cumsum() for the running total field



ACL command: LOOP

Executes a series of ACLScript commands repeatedly on a record while a specified condition evaluates to true

```
# Step 7/ Add any missing dates to the list of exchange rates
#7.1/ Create a list of From and To currency pairs
# - For each From and To currency pair that is found in D21_01_TT_DF_TCURR (here copied to D21_03_TT_DF_FCURRENTRY_CURRENT)
for (ZV_TCURRENTRY_CURRENT, ZV_TCURRENTRY_CURRENT) in D21_01_TT_DF_TCURR.groupby(['FCURRENTRY', 'TCURRENTRY']):
    #7.2/ Create an array with the FULL list of dates between the min and max date for the currency pair
    ZV_AR_FCURRENTRY_GDATU_FULL = PI_PANDAS.date_range(start=D21_03_TT_DF_FCURRENTRY_CURRENT['ZF_TCURRENTRY_GDATU'].min(),
    end=D21_03_TT_DF_FCURRENTRY_CURRENT['ZF_TCURRENTRY_GDATU'].max())
    #7.3/ Create a dataframe that has the full list of dates in a column
    D21_04_TT_DF_FCURRENTRY_GDATU_FULL = PI_PANDAS.DataFrame(ZV_AR_FCURRENTRY_GDATU_FULL, columns=['ZF_TCURRENTRY_GDATU_ALL'])
    #7.4/ Add the From currency and To currency fields to the full list of dates
    D21_05_TT_DF_GDATUFULL_ADD_UKURSF_CURRENTRY = PI_PANDAS.merge(D21_04_TT_DF_FCURRENTRY_GDATU_FULL,
    D21_03_TT_DF_FCURRENTRY_CURRENT, left_on='ZF_TCURRENTRY_GDATU_ALL', right_on='ZF_TCURRENTRY_GDATU', how='left')
    #7.5/ Copy the exchange rate from the line above, if it is NULL (due to missing date)
    D21_05_TT_DF_GDATUFULL_ADD_UKURSF_CURRENTRY['ZF_TCURRENTRY_UKURS'] = D21_05_TT_DF_GDATUFULL_ADD_UKURSF_CURRENTRY['ZF_TCURRENTRY_UKURS'].
    fillna(method='ffill').fillna(method='bfill')
    #7.6/ Enter the from and To currencies from the variables
    D21_05_TT_DF_GDATUFULL_ADD_UKURSF_CURRENTRY['ZF_TCURRENTRY_FCURRENTRY'] = ZV_TCURRENTRY_CURRENT
    D21_05_TT_DF_GDATUFULL_ADD_UKURSF_CURRENTRY['ZF_TCURRENTRY_TCURRENTRY'] = ZV_TCURRENTRY_CURRENT
    #7.7/ Append the data for the current currency pair into the final table
    D21_02_RT_DF_TCURRENTRY_ADDMSGDATES = PI_PANDAS.concat([D21_02_RT_DF_TCURRENTRY_ADDMSGDATES, D21_05_TT_DF_GDATUFULL_ADD_UKURSF_CURRENTRY
    [['ZF_TCURRENTRY_GDATU_ALL', 'ZF_TCURRENTRY_FCURRENTRY', 'ZF_TCURRENTRY_TCURRENTRY', 'ZF_TCURRENTRY_UKURS']]], ignore_index=True)

# Step 8/ Write the final DataFrame to a CSV file and print it
D21_02_RT_DF_TCURRENTRY_ADDMSGDATES.to_csv(ZV_RESULTS_FOLDER + '\D21_02_RT_DF_TCURRENTRY_ADDMSGDATES.csv', sep='\t', index=False)
print(D21_02_RT_DF_TCURRENTRY_ADDMSGDATES)
```

Explanation of python script

- Loop to fill in any missing dates in the list of exchange rates table (TCURRENTRY)
- Using date_range to get all the dates between the min and the max for the currency key combinations



A naming convention for Python



A naming convention for Python

It helps to have a naming convention!



PythonNamingCo
nvention



**Naming conventions enable you to be able to read the code
from your team much quicker**



**What else would you
like to do in Python?**



A naming convention for Python

Any other things that you need to do in Python?

Let us know in the *chat* what you think

What would be interesting to do in Python?





Extra poll question

Do you require a CPE certificate for joining today?

Do you require a CPE certificate for joining today?



(You will receive an email from **info@auditopia.com** , if you respond Yes, in order to take the Quiz for your certificate.)

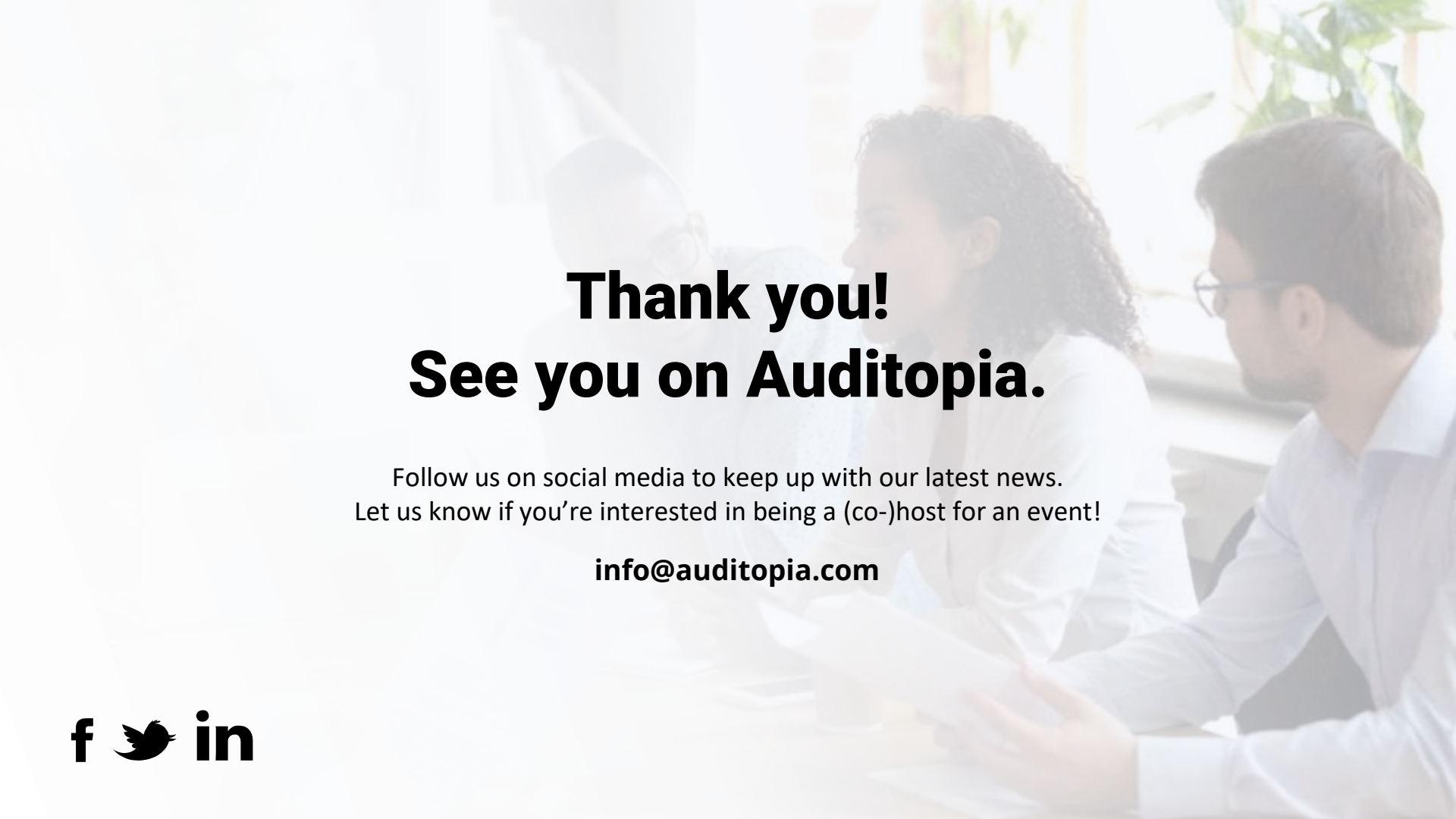


Please share your feedback

Please take the survey and let us know your feedback on the class!

<https://shorturl.at/guwSJ>





Thank you! **See you on Auditopia.**

Follow us on social media to keep up with our latest news.
Let us know if you're interested in being a (co-)host for an event!

info@auditopia.com

