Data Analysis Tests for IT Auditors

Michael Kano

Core Competencies – C33

Agenda

- Why use DA tools like Arbutus Analyzer?
- Pre-Testing Considerations
- 3 Typical Test Scenarios
- Open Discussion

Why use Arbutus?

- Reads most data types and formats
- Processes up to 100,000 records/second
- Strong analytic features
- Ease of use
- Applications beyond financial audits

Pre-Testing Considerations

- Data Integrity Verification
  - Blanks
  - Invalid content
  - Uniqueness
- Harmonization/normalization of key fields
  - Case
  - Length
  - Content

Data Integrity Verification

- Blanks/Zeros
- Invalid content
- Uniqueness

Blanks/Zeros

Data Type	Commands
Character	COUNT IF ISBLANK(*field*)
Date/Numeric	STATISTICS ON *field*
Numeric	COUNT IF *field* = 0

Blank Test Results

```
@ COUNT IF ISBLANK(Trans_Type)
```

Count	8,571

```
  8571 of 876401 met the test: ISBLANK(Trans_Type)
```

Zeros

- STATISTICS ON numeric field
- COUNT IF *numeric field* = 0

Zero Test Results: STATISTICS

```
@ STATISTICS ON Trans_Amount NUMBER 5
```

Field: Trans_Amount	Number	Total	Average
Positive	876,387	4,379,919,780.54	4,997.70
Zeros	2		
Negative	12	-60,310.83	-5,025.90
Totals	876,401	4,379,859,469.71	4,997.55
Abs Value		4,379,980,091.37	
Range		19,600.06	

Highest	Lowest
9,999.99	-9,600.07
9,999.99	-8,318.18
9,999.94	-8,072.84
9,999.94	-7,073.64
9,999.94	-6,176.50

Zero Test Results: COUNT IF…

```
@ COUNT IF Trans_Amount = 0
```

Count	2

```
  2 of 876401 met the test: Trans_Amount = 0
```

Blank/Invalid Dates

- STATISTICS ON *date field*

```
@ STATISTICS ON Trans_Date NUMBER 5
```

Field: Trans_Date	Number	Total	Average
Valid	876,384		12/02/2013
Invalid	9		
Blank	8		
Weekends	251,553		
Range		369 Days	

Highest	Lowest
06/05/2014	06/01/2013
06/05/2014	06/01/2013
06/05/2014	06/01/2013
06/05/2014	06/01/2013
06/05/2014	06/01/2013

Invalid Content

- Invalid character entry, e.g., transaction code
- Incorrect character data format
- Invalid date entry, e.g., 4/33/2014
- Negative amounts

Character Data: Invalid Code

- CLASSIFY ON *field*

```
@ CLASSIFY ON Trans_Type
```

Trans_Type	Count	<-- %
_	8571	0.98%
106	253579	28.93%
135	404079	46.11%
147	93155	10.63%
188	98579	11.25%
999	18438	2.10%
	876401	100.00%

```
6 records produced
```

Character Data: Character Content

- Key character required
  - Email address
  - Use COUNT IF NOT(FIND("@",*email field*)

```
@ COUNT IF NOT FIND("@",Email)
```

Count	573

573 of 876401 met the test: NOT FIND("@",Email)

Character Data: Format

- Consistent format required
  - SSN, transaction codes, phone numbers, zip/postal codes
  - Use CLASSIFY ON FORMAT(*character field*)

```
@ CLASSIFY ON FORMAT(SSN)
```

FORMAT(SSN)	Count	<-- %
999 99 9999	6	0.00%
999/99/9999	9	0.00%
99999999	876386	100.00%
	876401	100.00%

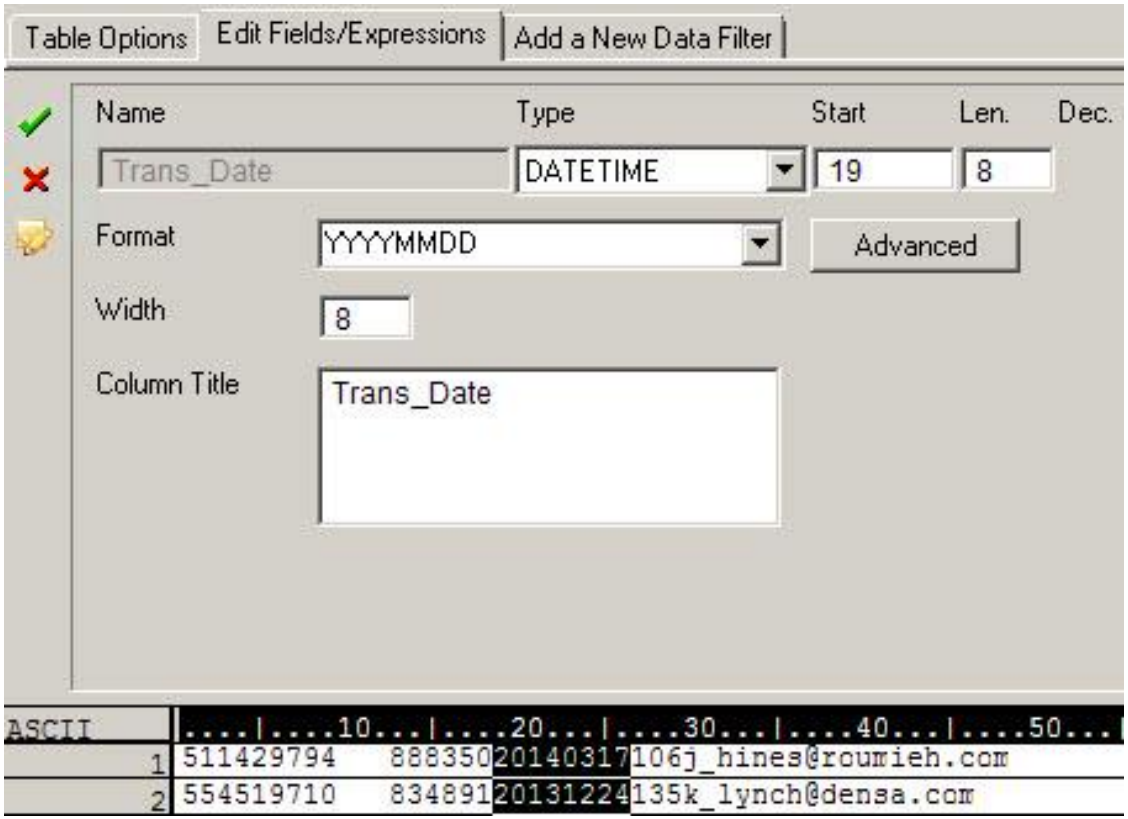
```
3 records produced
```

Managing Invalid Data

- Check field definition for errors
- Exclude records by filtering
- Harmonize data

Check Field Definition

- Edit>>Table Layout
- Usually date-time fields



Filtering

- EXTRACT with filter to new table

- *EXTRACT RECORD TO Transactions_2 IF Trans_date >= `20140101`*

Data Harmonization Functions

- Content
  - INCLUDE()
  - EXCLUDE()
  - REPLACE()
  - INSERT()
- Length
  - SUBSTRING()
- Case
  - UPPER()
  - LOWER()

Example: SSN Field

- Objective: SSN field in *nnn-nn-nnnn* format
- Create computed field SSN_Clean with nested functions
- Step 1: Remove non-numeric characters
  INCLUDE(SSN,"1~0")
- Step 2: Insert first hyphen:
  INSERT(INCLUDE(SSN,"1~0"),4,"-")
- Step 3: Insert second hyphen:
  INSERT(INSERT(INCLUDE(SSN,"1~0"),4,"-"),7,"-")

Example: SSN Field

Step	SSN
Original	062/33/1234
Step 1	062331234
Step 2	062-331234
Step 3	062-33-1234

IT Audit Tests

- User Access
- Data Migration
- Keyword Search

User Access

- Compare access list to current employees
- Access list includes login ID and name
- Current employee list includes login ID, first name, and last name
- System login ID not necessarily identical to employee list login ID

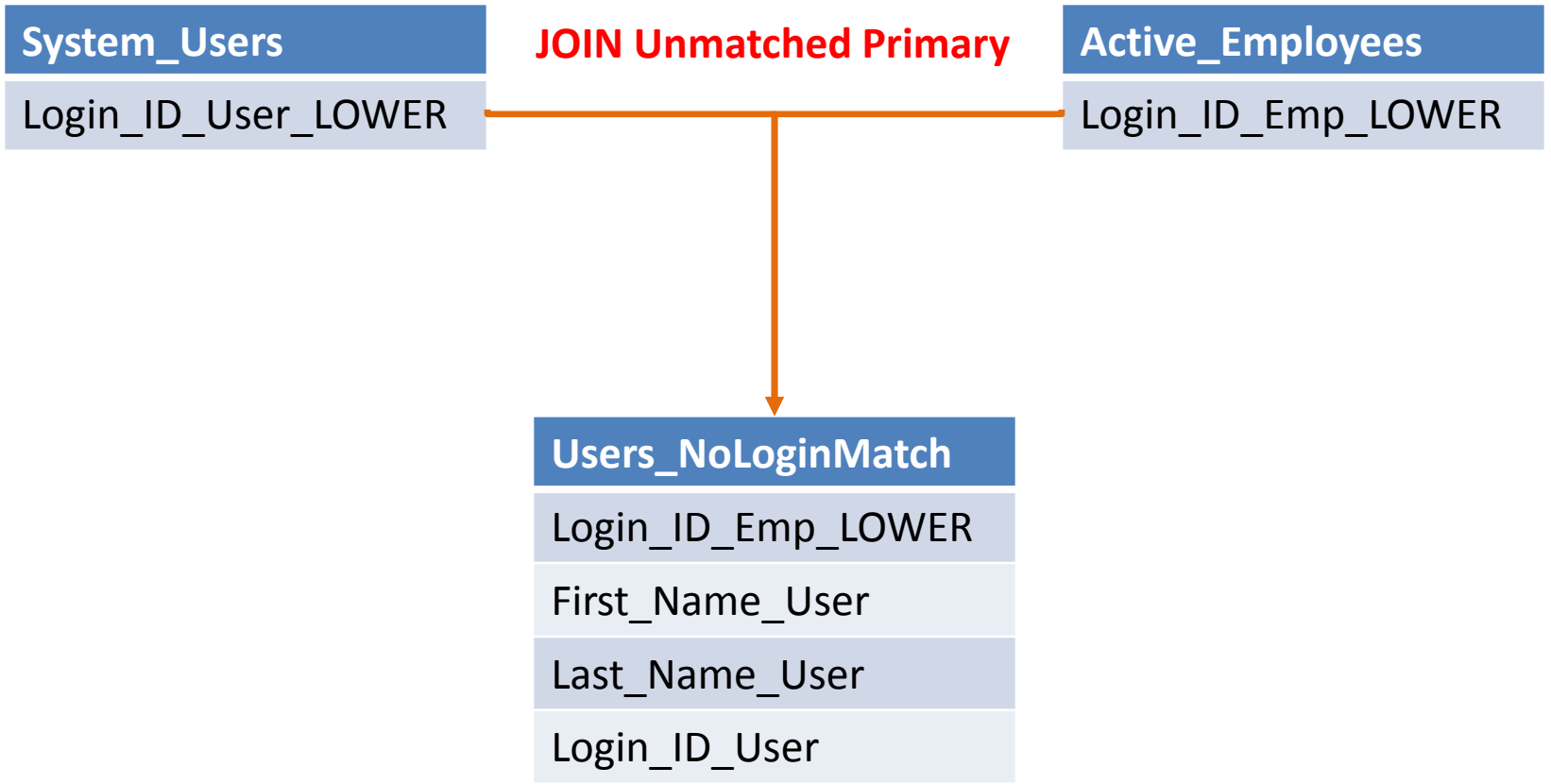
User Access: Data Analysis Plan

- Harmonize key fields
- Compare login IDs
- If no match, compare names
- Use JOIN command

User Access: JOIN on Login IDs

- Convert both login ID fields to lower-case with LOWER() function
- Execute JOIN UNMATCHED from access list using active employee list as secondary
- Result will contain all users who are not in active employee list

User Access: Join on Login IDs



User Access: Login ID JOIN Results

```
@ OPEN System_Users

@ OPEN Active_Employees SECONDARY

@ JOIN UNMATCHED PKEY Login_ID_User_LOWER
FIELDS ALL SKEY Login_ID_Emp_LOWER TO
"Users_Unmatched_Login_ID" OPEN PRESORT
SECSORT
Presorting Primary data file.
Presorting Secondary data file.
  208 records produced
  1107 records bypassed
```

User Access: Compare on Names

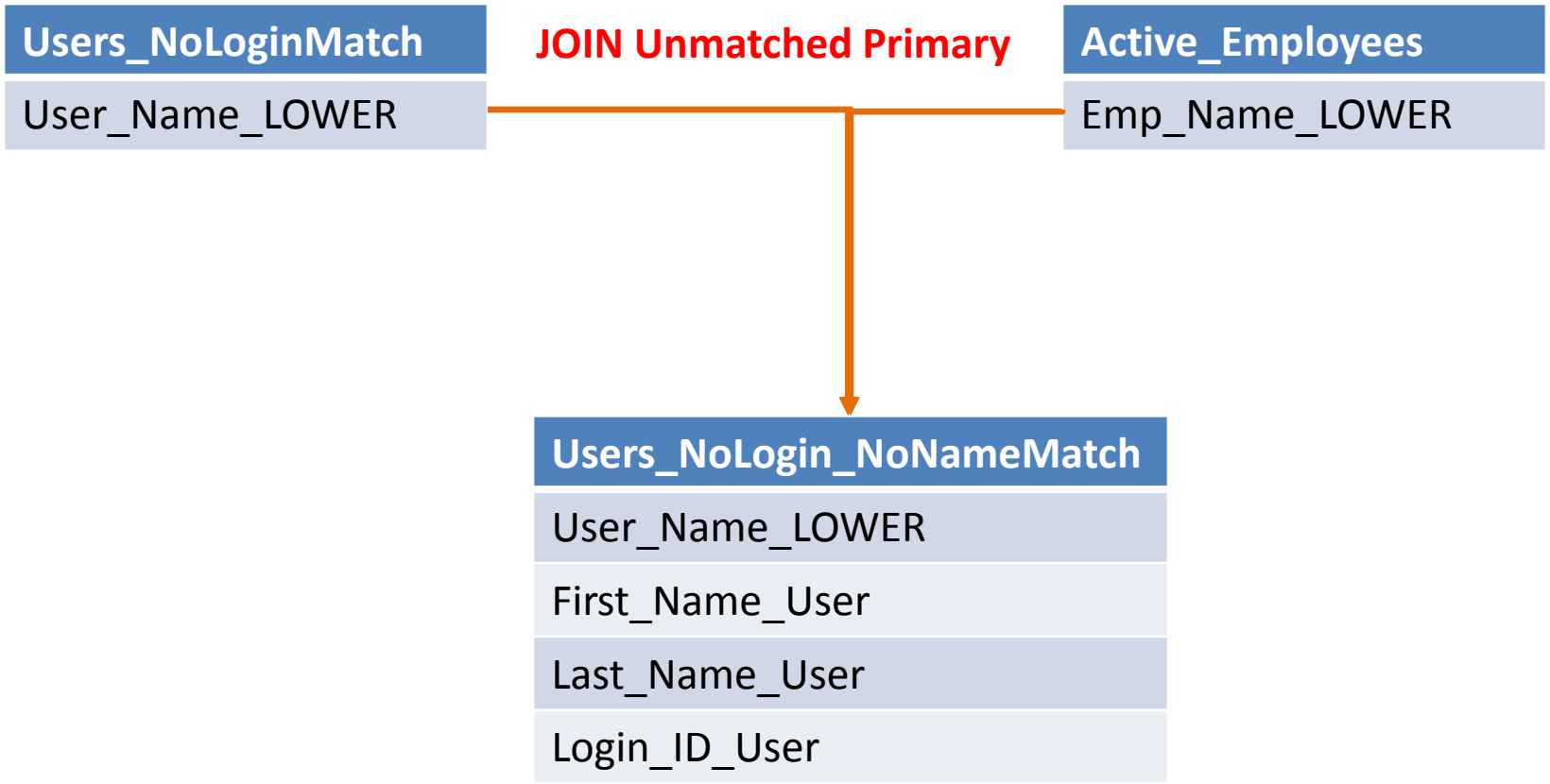
- Harmonize and concatenate name fields
- Execute JOIN UNMATCHED from unmatched access records using active employee list as secondary
- Result will contain all users that do not match on login ID or on name

User Access: Harmonize Names

- Create computed field Name_Harmonize
  LOWER(ALLTRIM(first_name)) +
  LOWER(ALLTRIM(last_name))

	First_Name	Last_Name	Emp_Name_LOWER
1	Amy	Austin	amyaustin
2	Alisa	Bellis	alisabellis
3	Arleen	Bergert	arleenbergert
4	Alma	Bowman	almabowman
5	Ada	Bradley	adabradley
6	Alma	Brunner	almabrunner

User Access: Join on Login IDs



User Access: Name JOIN Results

```
@ OPEN Unmatched_Login_ID

@ OPEN AEL_DEL SECONDARY

@ JOIN UNMATCHED PKEY User_Name_LOWER FIELDS
ALL SKEY Emp_Name_LOWER TO
"Unmatched_LoginID_Name" OPEN PRESORT
SECSORT
Presorting Primary data file.
Presorting Secondary data file.
   1 records produced
   207 records bypassed
```

User Access: JOIN on Name

- Can create tighter JOIN by restricting computed fields to alphabetic characters using INCLUDE(field,"a~z")
- Can test for last name match only and refine further
- Can add fuzzy duplicate component to test

Keyword Search

- Useful for log/memo testing
  - IT Tickets
  - Customer service
- Need master keyword list

Keyword List Search

- Requires text file containing keywords
- One word/phrase per line
- Not case-sensitive
- Can edit list
- Can look up >1 list

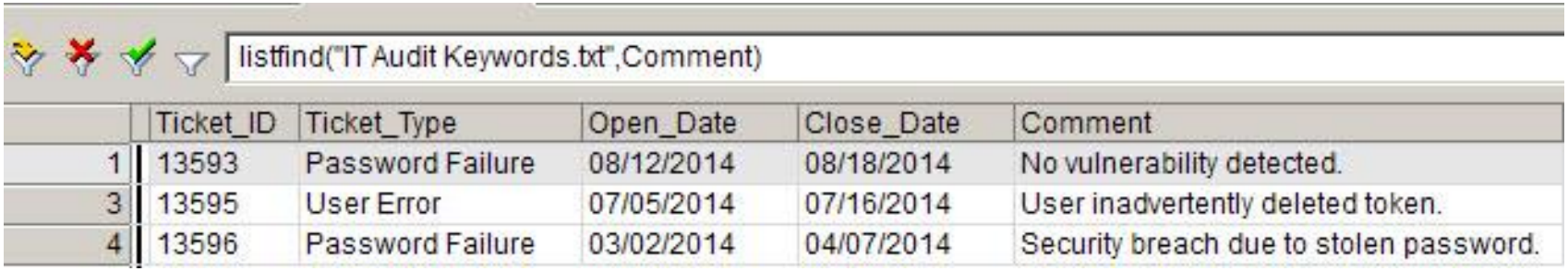
Use LISTFIND() Function

Operation	List file(s)	Fields	Syntax
Finding one list of items in a record	Tech.txt		LISTFIND("Tech.txt")
Finding one list of items in a field	Tech.txt	Comment	LISTFIND("Tech.txt", Comment)
Finding one list of items in multiple fields	Tech.txt	Comment, Message	LISTFIND("Tech.txt", Comment, Message)
Finding multiple lists of items in one field	Tech.txt AML.txt	Comment	LISTFIND("Tech.txt, AML.txt", Comment)
Finding multiple lists of items in multiple fields	Tech.txt AML.txt	Comment, Message	LISTFIND("Tech.txt, AML.txt", Comment, Message)

Use LISTFIND() in a Filter

- In a view filter: display records that meet the criteria
- With EXTRACT command to write exceptions to new table:
  - *EXTRACT IF LISTFIND(…) TO file name*

Keyword Search Results



	Ticket_ID	Ticket_Type	Open_Date	Close_Date	Comment
1	13593	Password Failure	08/12/2014	08/18/2014	No vulnerability detected.
3	13595	User Error	07/05/2014	07/16/2014	User inadvertently deleted token.
4	13596	Password Failure	03/02/2014	04/07/2014	Security breach due to stolen password.

Keyword Search Script

Can identify:

- Field(s) in which keywords occur
- Position in field where keywords occur (use AT() function)
- Number of times keywords occur (use CLASSIFY command)

Extra Credit: Data Migration

- Useful for validating internal data processing
- Also for data migration to new system
- Verify that all data transferred with no unintended transformation or truncation
- Run data integrity tests on each file prior to migration testing

Data Migration: Identical Fields

- Use *DISPLAY PRIM TO table_TL* on each file
- Writes table layouts to tables
- Use JOINs on field names to match fields and compare lengths, types, etc…

Data Migration: Compare Layouts

- If layouts do not match in name, type, length, review with data providers
- If layouts match, go on to high-level comparisons

Data Migration: High-Level Tests

- Record count (COUNT)
- Control/hash totals (TOTAL/STATISTICS)
- STATISTICS on date fields
- JOIN on unique record identifier
- CLASSIFY on text fields and compare results using JOINs
- SUMMARIZE on date field and compare results using JOINs

Data Migration: Granular Tests

- JOIN on unique record identifier
- CLASSIFY on text fields and compare results using JOINs
- SUMMARIZE on date field and compare results using JOINs

Data Migration: Addressing Errors

Type	Try…	Comment
Record counts don't match	Run DUPLICATES command on unique record identifier in larger file.	
	Run JOIN command on unique record identifier.	
Numeric totals don't match	Run CLASSIFY on text field and ACCUMULATE the numeric field.	This will help you identify the particular class of record(s) where the error might have originated.
	Run STRATIFY on the numeric field in both files and compare the results by strata.	Determine if particular transaction size as a source of error or decimal point misplaced.
	Run CLASSIFY on Year-Month field and accumulate the numeric field.	Determine if period of transaction is the source of error.

Any questions?

Michael Kano

michaeldk@telus.net