Desensitizing Non-Production

Reducing Security Concerns in Test Environments

John Haldeman – Hackforge – June 2017

Sensitive and why it’s important

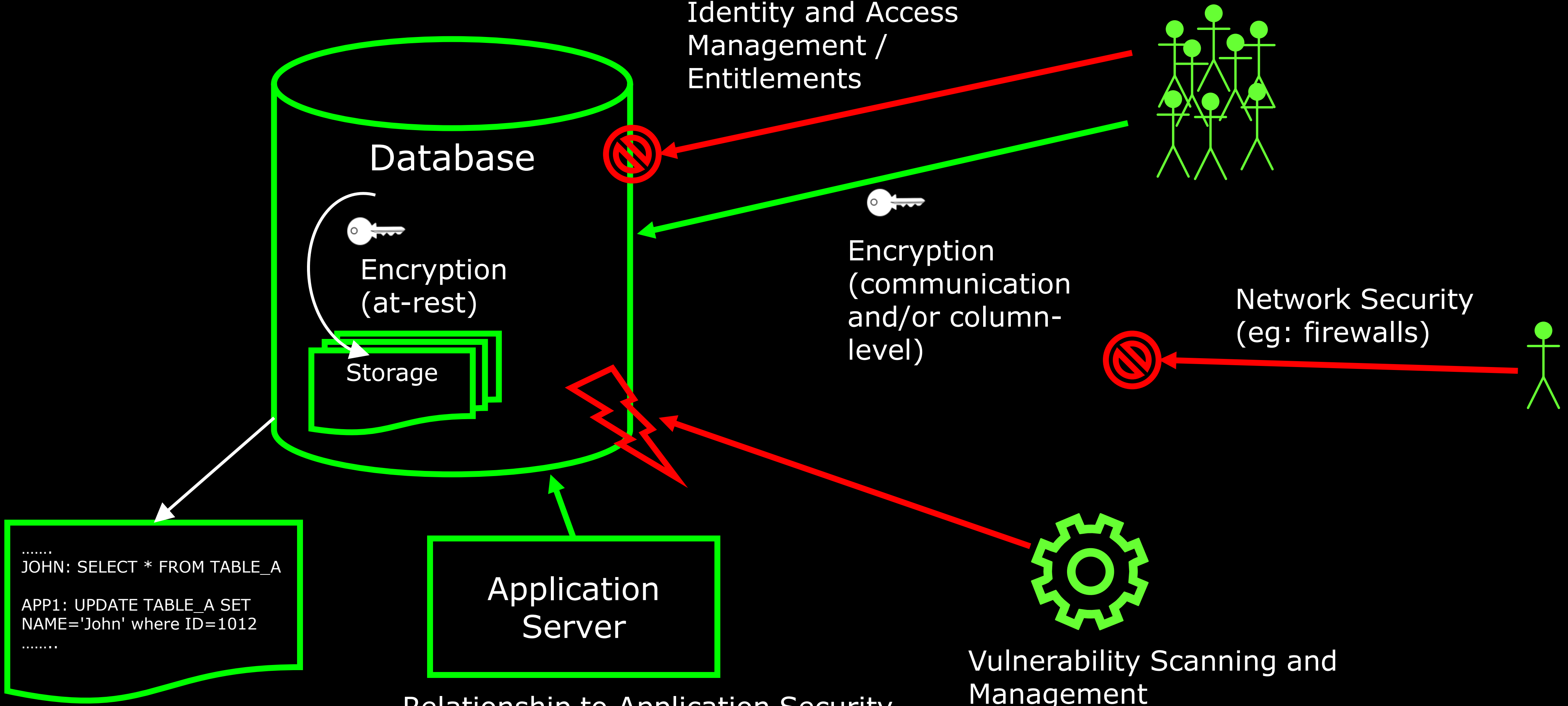
It’s Like… Desensitizing Paste but for Test Environments!*

- Database Security at a Glance
- Managing Risks
- Security and Test Environments
- Masking Mechanics
- Commercial Packages
- Challenges

Colgate
Sensitive
Pro-Relief
DESENSITIZING PASTE / PÂTE DÉSENSIBILISANTE
IN OFFICE TREATMENT / TRAITEMENT AU CABINET

* This is why John isn’t in marketing

Database Security at a Glance


Identity and Access Management / Entitlements
Database
Encryption (at-rest)
Storage
Encryption (communication and/or column-level)
Network Security (eg: firewalls)
Application Server
Vulnerability Scanning and Management

```
…….
JOHN: SELECT * FROM TABLE_A

APP1: UPDATE TABLE_A SET
NAME='John' where ID=1012
……..
```

Audit Logging

Relationship to Application Security (eg: SQL Injection Prevention, Application Identity Management, trusted contexts)

Test Environments

Goal: Allow software developers and testers to look for software defects

Looking for defects tends to require a lot of ***flexibility***

Implementing production-level security controls in test reduces flexibility and increases test environment cost – most people leave them out

Managing Risk

- Two factors affecting risk:
  - The likelihood of occurrence
  - The cost of occurrence

- Reducing risk
  - Decrease the likelihood of occurrence
  - Decrease the cost of occurrence

Test Environments

Reducing Security Risks in Test: Tend to focus on decreasing the cost of occurrence rather than the likelihood

Reduce the cost of occurrence:

Use dummies (“masked” data) not real people (sensitive data) – Not a perfect analogy but you get the idea

Building Crash Test Dummies

Criteria (I assume):

1) Not a real person
2) Similar enough to a real person for the test to be effective

Design (I assume):

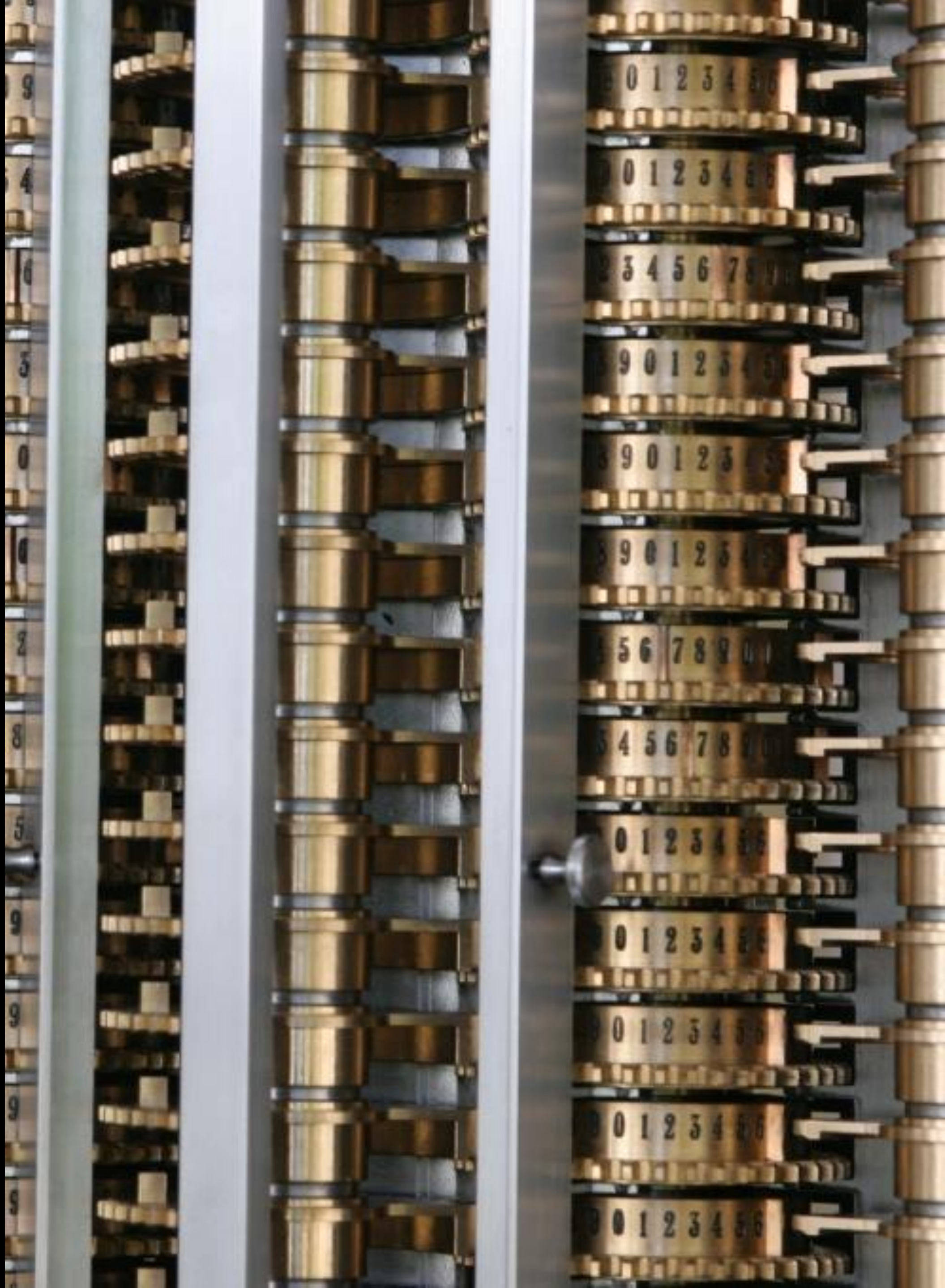
1) Take a real person
2) Make a model similar to it

Test Data “Desensitization” (or masking) does something similar:

1) Take real data
2) Manipulate it until it’s no longer sensitive
3) Don’t break it’s usefulness for testing during processing

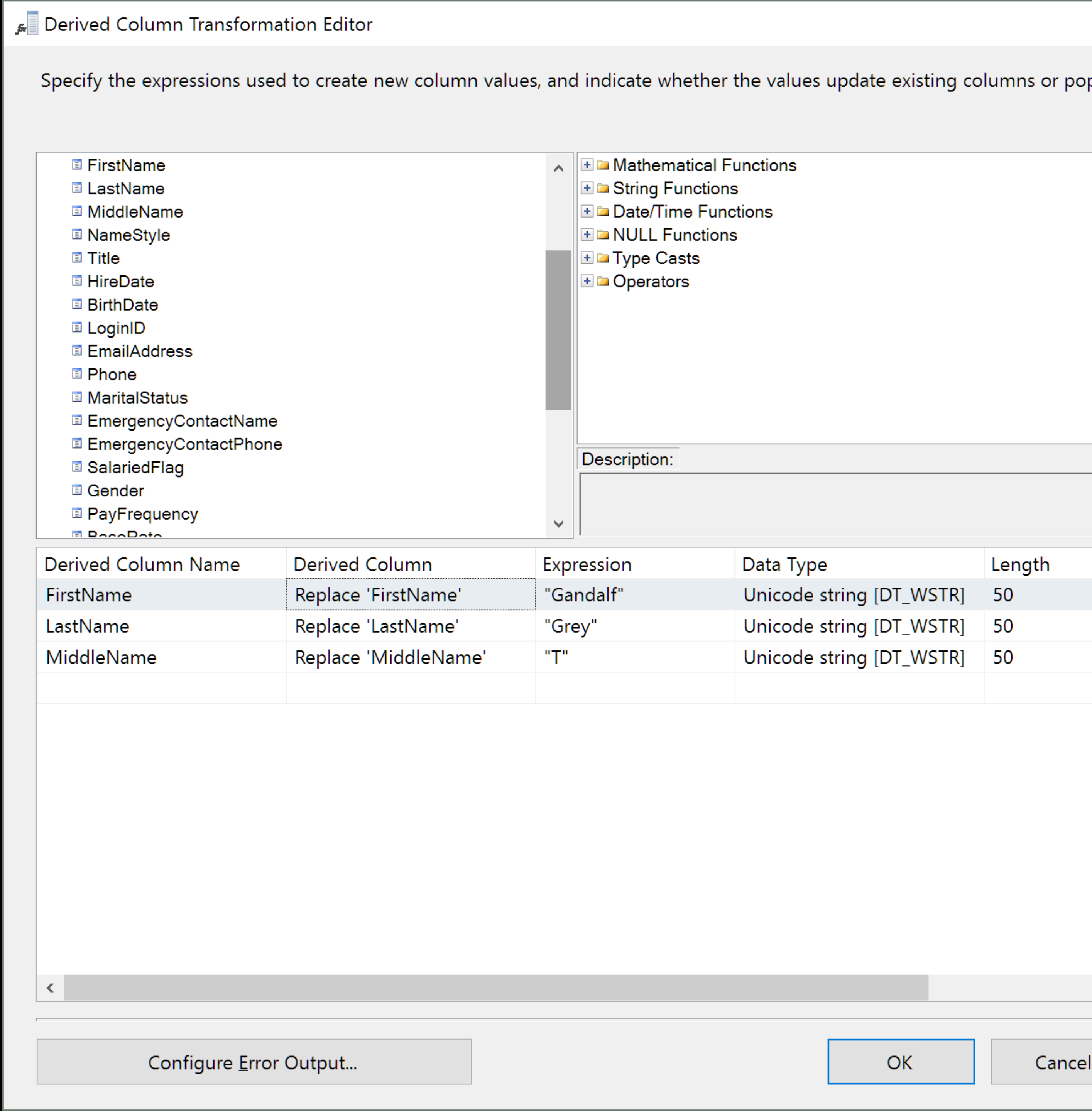
Masking Mechanics

- Literal Replacement
- Random/Hashed Algorithmic Replacement
- Random/Hashed Lookups
- Shuffling



Literal Replacement

- Easy enough: *One String to rule them all*
- Destroys the data – very secure and very easy
- *Problem:* Does not create very realistic test cases



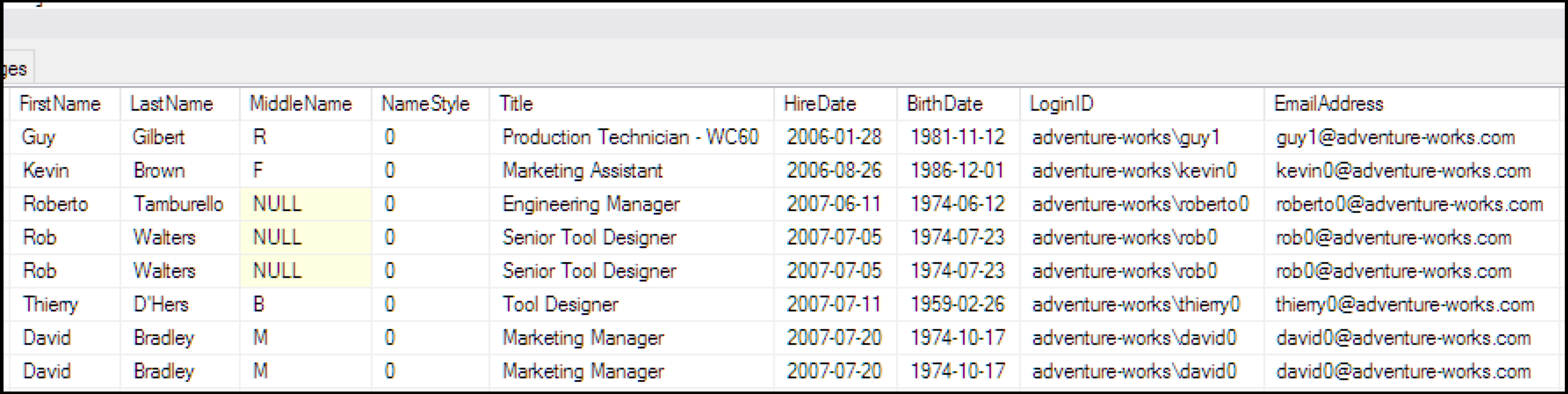
Derived Column Transformation Editor

Specify the expressions used to create new column values, and indicate whether the values update existing columns or pop

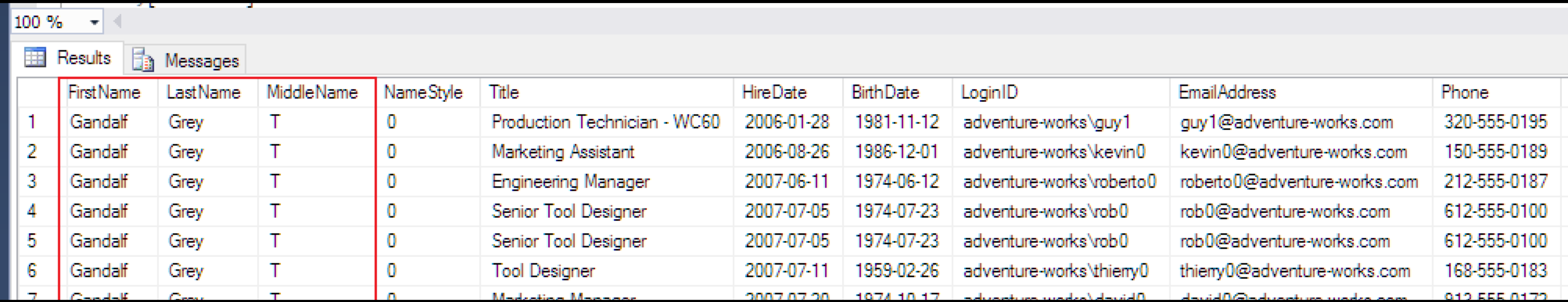
- FirstName
- LastName
- MiddleName
- NameStyle
- Title
- HireDate
- BirthDate
- LoginID
- EmailAddress
- Phone
- MaritalStatus
- EmergencyContactName
- EmergencyContactPhone
- SalariedFlag
- Gender
- PayFrequency

Derived Column Name	Derived Column	Expression	Data Type	Length
FirstName	Replace 'FirstName'	"Gandalf"	Unicode string [DT_WSTR]	50
LastName	Replace 'LastName'	"Grey"	Unicode string [DT_WSTR]	50
MiddleName	Replace 'MiddleName'	"T"	Unicode string [DT_WSTR]	50

Literal Replacement



FirstName	LastName	MiddleName	NameStyle	Title	HireDate	BirthDate	LoginID	EmailAddress
Guy	Gilbert	R	0	Production Technician - WC60	2006-01-28	1981-11-12	adventure-works\guy1	guy1@adventure-works.com
Kevin	Brown	F	0	Marketing Assistant	2006-08-26	1986-12-01	adventure-works\kevin0	kevin0@adventure-works.com
Roberto	Tamburello	NULL	0	Engineering Manager	2007-06-11	1974-06-12	adventure-works\roberto0	roberto0@adventure-works.com
Rob	Walters	NULL	0	Senior Tool Designer	2007-07-05	1974-07-23	adventure-works\rob0	rob0@adventure-works.com
Rob	Walters	NULL	0	Senior Tool Designer	2007-07-05	1974-07-23	adventure-works\rob0	rob0@adventure-works.com
Thierry	D'Hers	B	0	Tool Designer	2007-07-11	1959-02-26	adventure-works\thierry0	thierry0@adventure-works.com
David	Bradley	M	0	Marketing Manager	2007-07-20	1974-10-17	adventure-works\david0	david0@adventure-works.com
David	Bradley	M	0	Marketing Manager	2007-07-20	1974-10-17	adventure-works\david0	david0@adventure-works.com



	FirstName	LastName	MiddleName	NameStyle	Title	HireDate	BirthDate	LoginID	EmailAddress	Phone
1	Gandalf	Grey	T	0	Production Technician - WC60	2006-01-28	1981-11-12	adventure-works\guy1	guy1@adventure-works.com	320-555-0195
2	Gandalf	Grey	T	0	Marketing Assistant	2006-08-26	1986-12-01	adventure-works\kevin0	kevin0@adventure-works.com	150-555-0189
3	Gandalf	Grey	T	0	Engineering Manager	2007-06-11	1974-06-12	adventure-works\roberto0	roberto0@adventure-works.com	212-555-0187
4	Gandalf	Grey	T	0	Senior Tool Designer	2007-07-05	1974-07-23	adventure-works\rob0	rob0@adventure-works.com	612-555-0100
5	Gandalf	Grey	T	0	Senior Tool Designer	2007-07-05	1974-07-23	adventure-works\rob0	rob0@adventure-works.com	612-555-0100
6	Gandalf	Grey	T	0	Tool Designer	2007-07-11	1959-02-26	adventure-works\thierry0	thierry0@adventure-works.com	168-555-0183
7	Gandalf	Grey	T	0	Marketing Manager	2007-07-20	1974-10-17	adventure-works\david0	david0@adventure-works.com	[illegible]

Algorithmic Replacement

Build a script to take the data and replace it with something that looks real

For example, this C# script generates a ***valid*** Random Canadian SIN number: 9 digit Luhn algorithm verified number

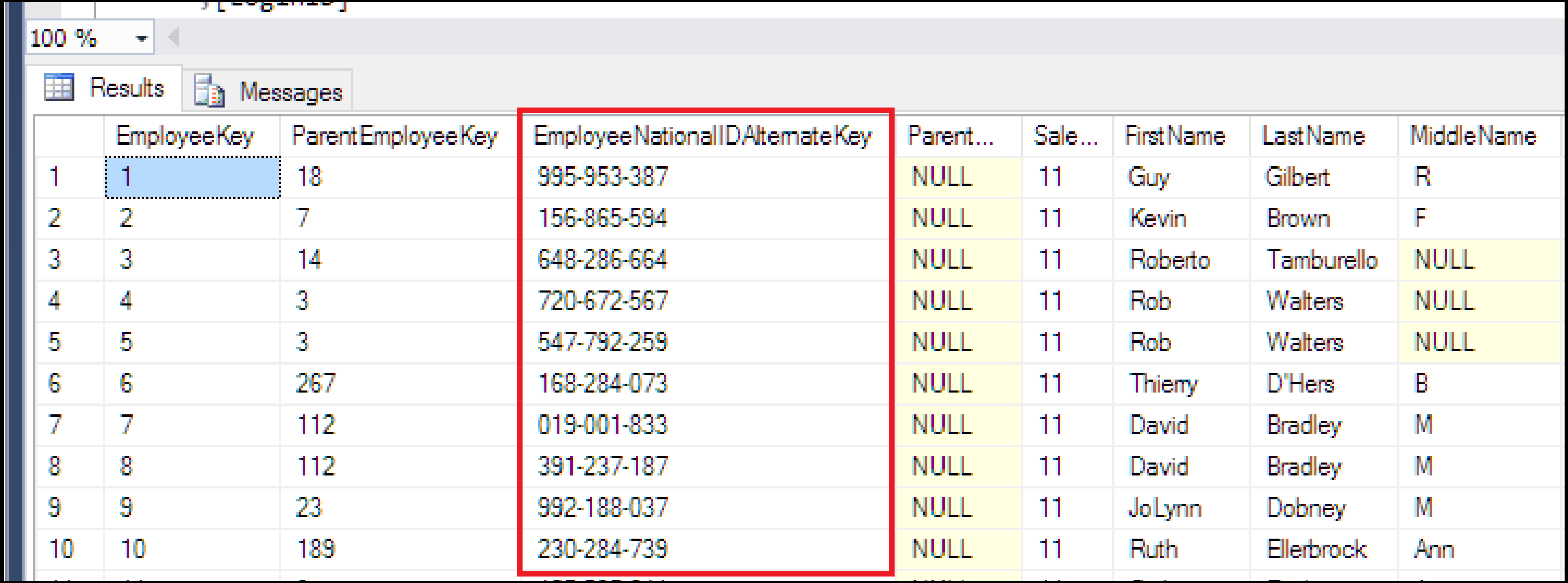
Common Credit Card PANs are 16 digit Luhn verified numbers with a limited set of prefixes

```csharp
public override void
         Input0_ProcessInputRow(Input0Buffer Row)
    {
        int[] sinNums = new int[8];
        int luhnTotal = 0;
        for(int i = 0; i < sinNums.Length; i++)
        {
            int newRand = this.rnd.Next(0, 10);
            sinNums[i] = newRand;
            if (i % 2 == 1) {
                int randDoubled = newRand * 2;
                if (randDoubled >= 10)
                    luhnTotal += randDoubled - 9;
                else
                    luhnTotal += randDoubled;
            }
            else {
                luhnTotal += newRand;
            }
        }

        int checkDigit = 10 - (luhnTotal % 10);
        if (checkDigit == 10)
            checkDigit = 0;

        Output0Buffer.AddRow();
        Output0Buffer.EmployeeKey = Row.EmployeeKey;
        Output0Buffer.EmployeeNationalIDAlternateKey =
           "" + sinNums[0] + sinNums[1] + sinNums[2] +
           "-" + sinNums[3] + sinNums[4] + sinNums[5] +
           "-" + sinNums[6] + sinNums[7] + checkDigit;
    }
```

Algorithmic Replacement - Results



	EmployeeKey	ParentEmployeeKey	EmployeeNationalIDAlternateKey	Parent...	Sale...	FirstName	LastName	MiddleName
1	1	18	995-953-387	NULL	11	Guy	Gilbert	R
2	2	7	156-865-594	NULL	11	Kevin	Brown	F
3	3	14	648-286-664	NULL	11	Roberto	Tamburello	NULL
4	4	3	720-672-567	NULL	11	Rob	Walters	NULL
5	5	3	547-792-259	NULL	11	Rob	Walters	NULL
6	6	267	168-284-073	NULL	11	Thierry	D'Hers	B
7	7	112	019-001-833	NULL	11	David	Bradley	M
8	8	112	391-237-187	NULL	11	David	Bradley	M
9	9	23	992-188-037	NULL	11	JoLynn	Dobney	M
10	10	189	230-284-739	NULL	11	Ruth	Ellerbrock	Ann

Using Hashes

Instead of random numbers hashes of other values can be used to generate the new number

```
public override void Input0_ProcessInputRow(Input0Buffer Row)
    {
        SHA256 sha2Hash = SHA256.Create();
        String saltySecret = "a397a7eaf79149abaa52f17241f680fb";
        byte[] hashdata =
          sha2Hash.ComputeHash(
          Encoding.UTF8.GetBytes(
                    Row.EmployeeNationalIDAlternateKey +
                    saltySecret));
        int[] sinNums = new int[8];
        int luhnTotal = 0;
        for(int i = 0; i < sinNums.Length; i++)
        {
            int newDecimal = hashdata[i] % 10;
            sinNums[i] = newDecimal;
            if (i % 2 == 1) {
                int randDoubled = newDecimal * 2;
                if (randDoubled >= 10)
                    luhnTotal += randDoubled - 9;
                else
                    luhnTotal += randDoubled;
            }
            else {
                luhnTotal += newDecimal;
            }
        }

        int checkDigit = 10 - (luhnTotal % 10);
        if (checkDigit == 10)
            checkDigit = 0;

        Output0Buffer.AddRow();
        Output0Buffer.EmployeeKey = Row.EmployeeKey;
        Output0Buffer.EmployeeNationalIDAlternateKey =
            "" + sinNums[0] + sinNums[1] + sinNums[2] +
            "-" + sinNums[3] + sinNums[4] + sinNums[5] +
            "-" + sinNums[6] + sinNums[7] + checkDigit;
    }
```

Why are hashes useful?

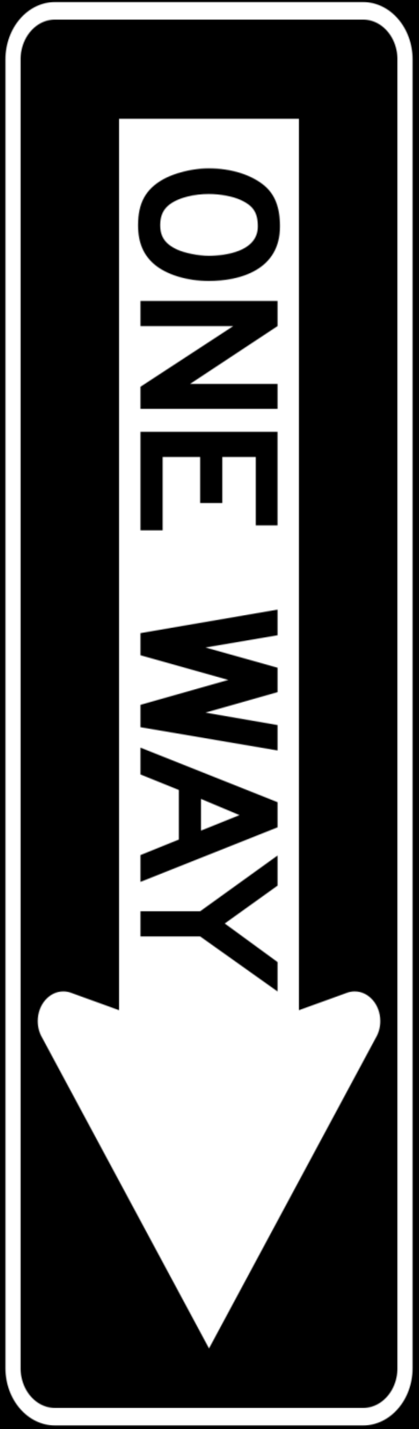
- Random looking but result in the same value being generated every run as long as the same key is provided
  - Prevents existing tests from being broken if you need to refresh the environment
  - Allows you to mask duplicated (non-normalized) data with the same values
  - Allows you to mask data between systems consistently

Security Problems with Hashes

Hashes are 1-way functions (which is good, you don't want people to reverse the values)

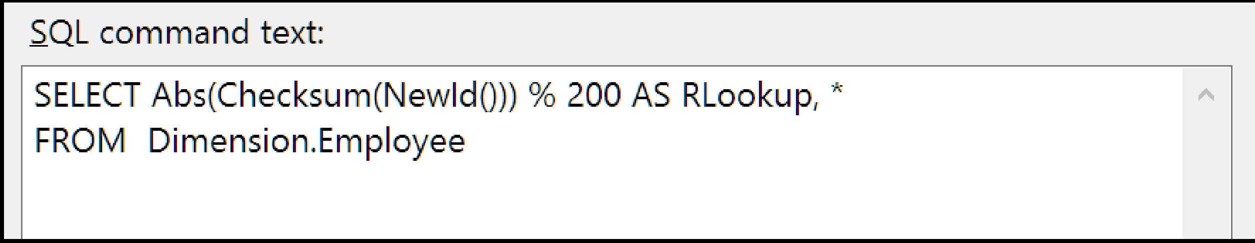
For small input domains (like SIN numbers for instance) you could relatively easily create a reverse lookup table if you knew the algorithm used (brute force) – Making the hash reversible

Using a Hash salt that you keep secret helps prevent this but then you have a secret to keep – randomly generated values do not require this



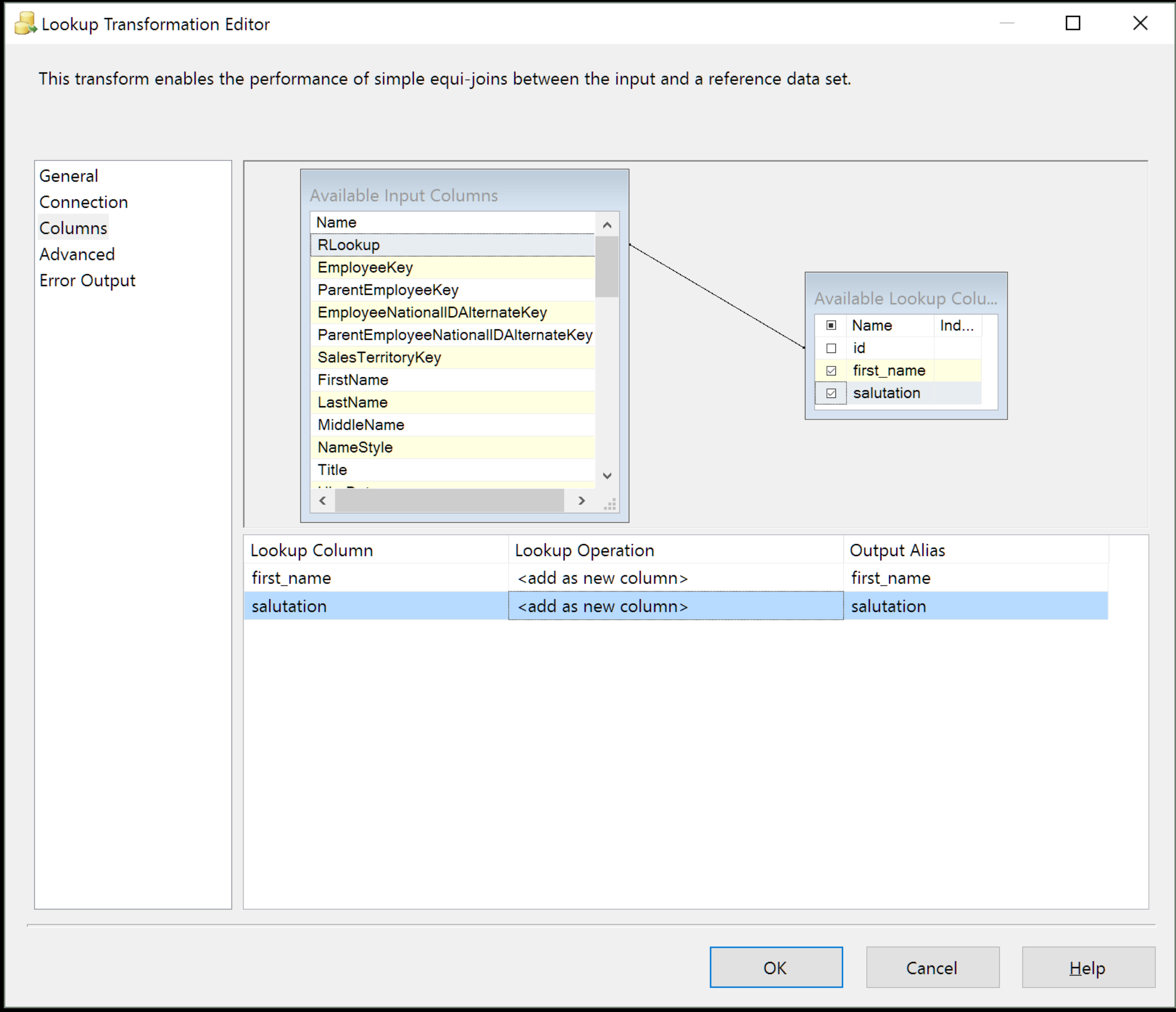
Lookups

- Some data elements are not easily (or practicably) generated by algorithm
- Use a lookup table instead!
- Addresses are a good example of this
- This is an example for randomly looking up first names (note how it replaces two columns so the first name and salutation are taken together)



SQL command text:

```
SELECT Abs(Checksum(NewId())) % 200 AS RLookup, *
FROM  Dimension.Employee
```



Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

Available Input Columns

Name
RLookup
EmployeeKey
ParentEmployeeKey
EmployeeNationalIDAlternateKey
ParentEmployeeNationalIDAlternateKey
SalesTerritoryKey
FirstName
LastName
MiddleName
NameStyle
Title

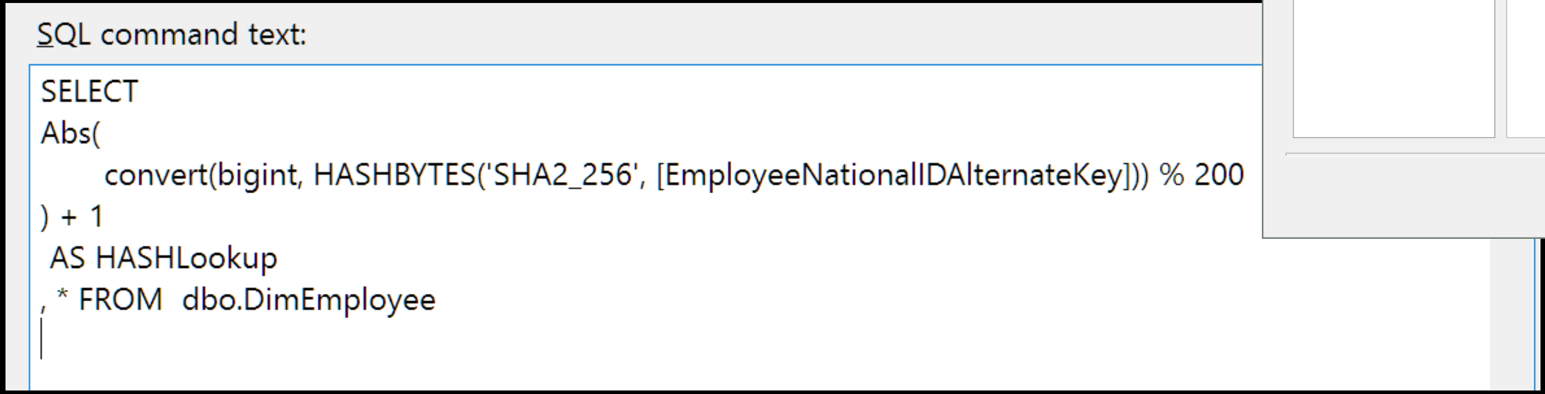
Available Lookup Colu...

	Name	Ind...
☐	id	
☑	first_name	
☑	salutation	

Lookup Column	Lookup Operation	Output Alias
first_name	<add as new column>	first_name
salutation	<add as new column>	salutation

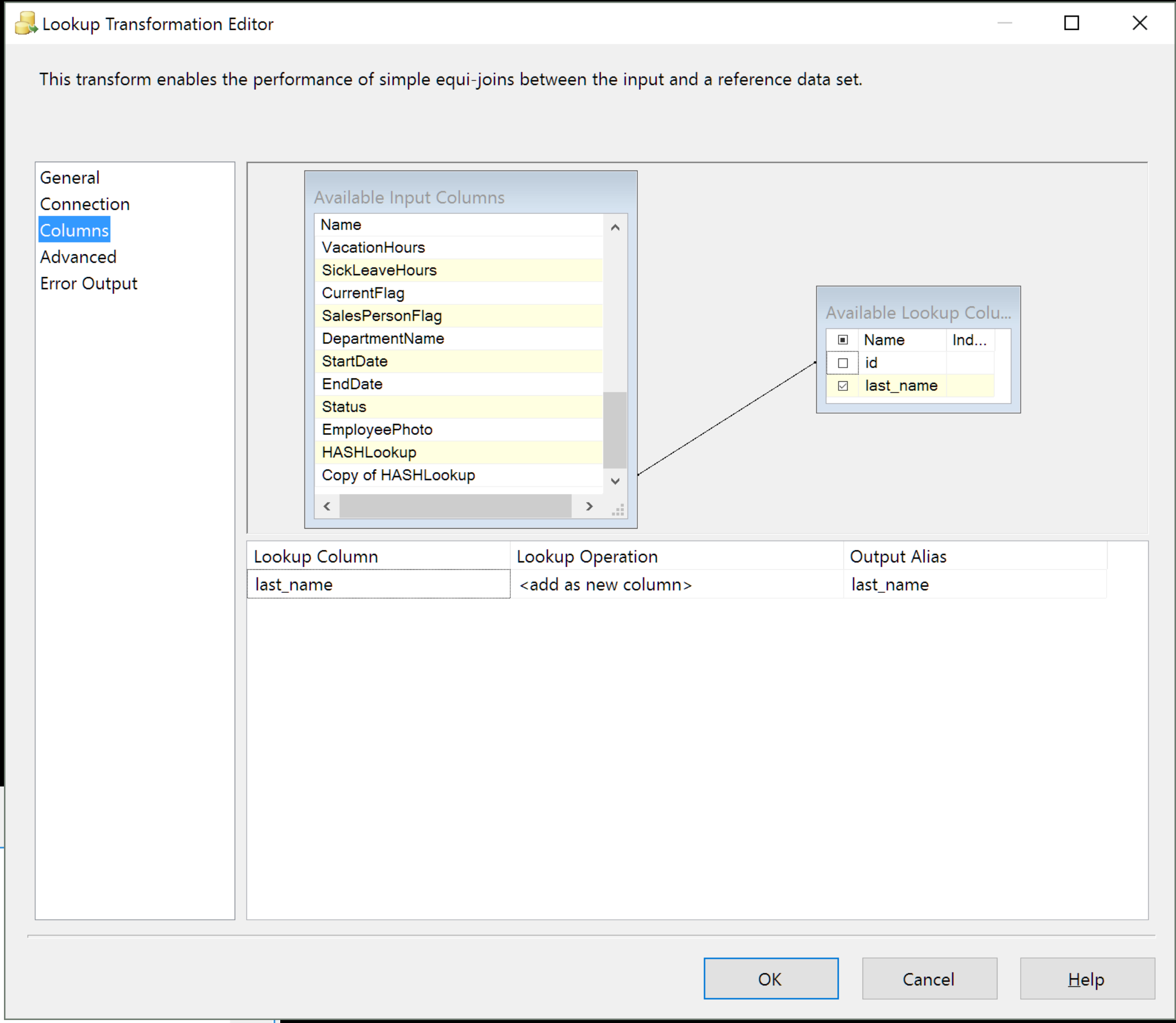
Hashed Lookups

- If you use a hashed value instead of a randomly generated value for the key, you can create a hashed lookup (for run-to-run consistency)



SQL command text:

```
SELECT
Abs(
    convert(bigint, HASHBYTES('SHA2_256', [EmployeeNationalIDAlternateKey])) % 200
) + 1
 AS HASHLookup
, * FROM  dbo.DimEmployee
```



Lookup Transformation Editor

This transform enables the performance of simple equi-joins between the input and a reference data set.

Available Input Columns

Name
VacationHours
SickLeaveHours
CurrentFlag
SalesPersonFlag
DepartmentName
StartDate
EndDate
Status
EmployeePhoto
HASHLookup
Copy of HASHLookup

Available Lookup Colu...

	Name	Ind...
☐	id	
☑	last_name	

Lookup Column	Lookup Operation	Output Alias
last_name	<add as new column>	last_name

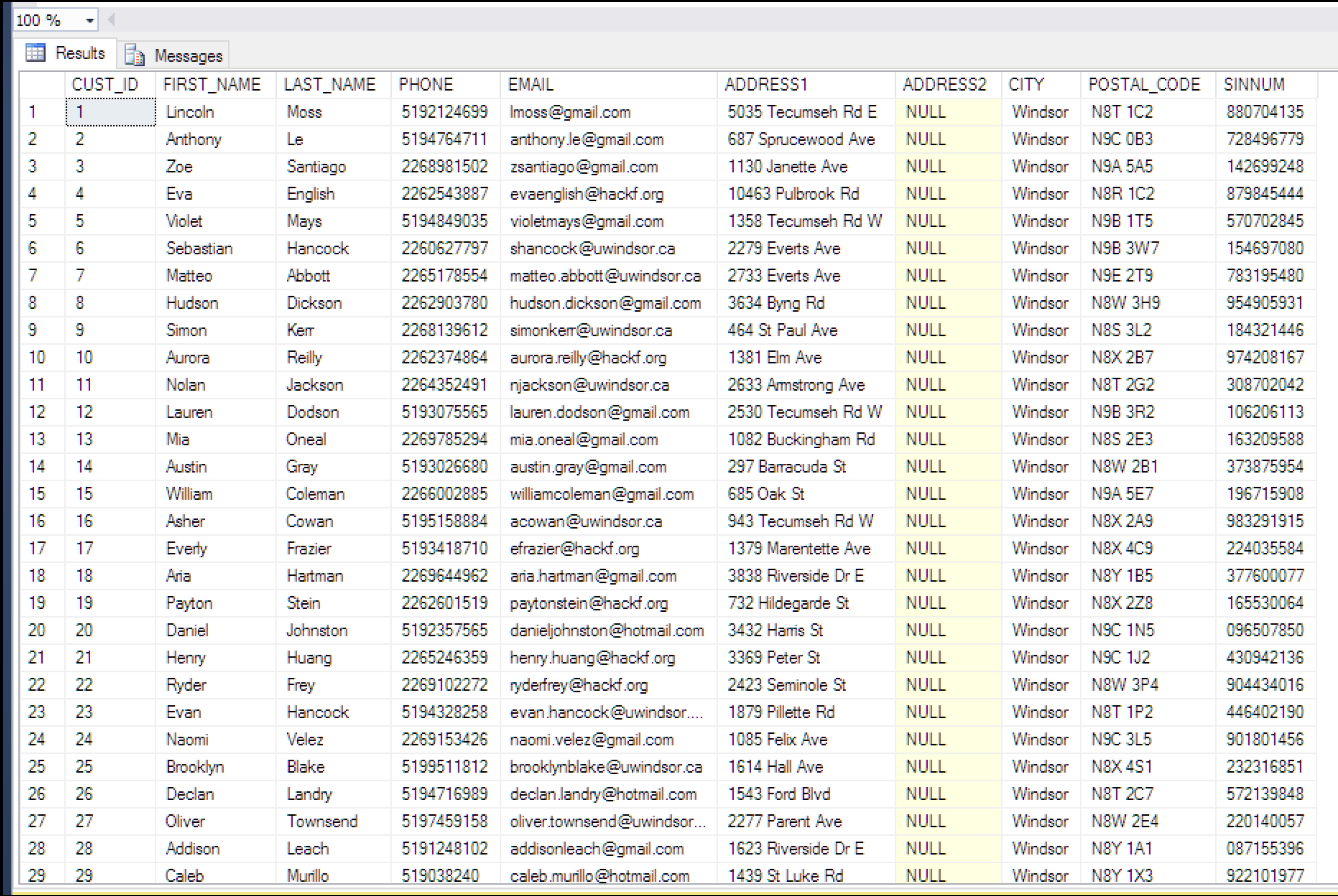
Shuffling

Mix up the sensitive values within a table instead of replacing them

One way to do this is with a random/hashed lookup on the same table and column you are masking (this shuffles in a way but may also duplicate data)

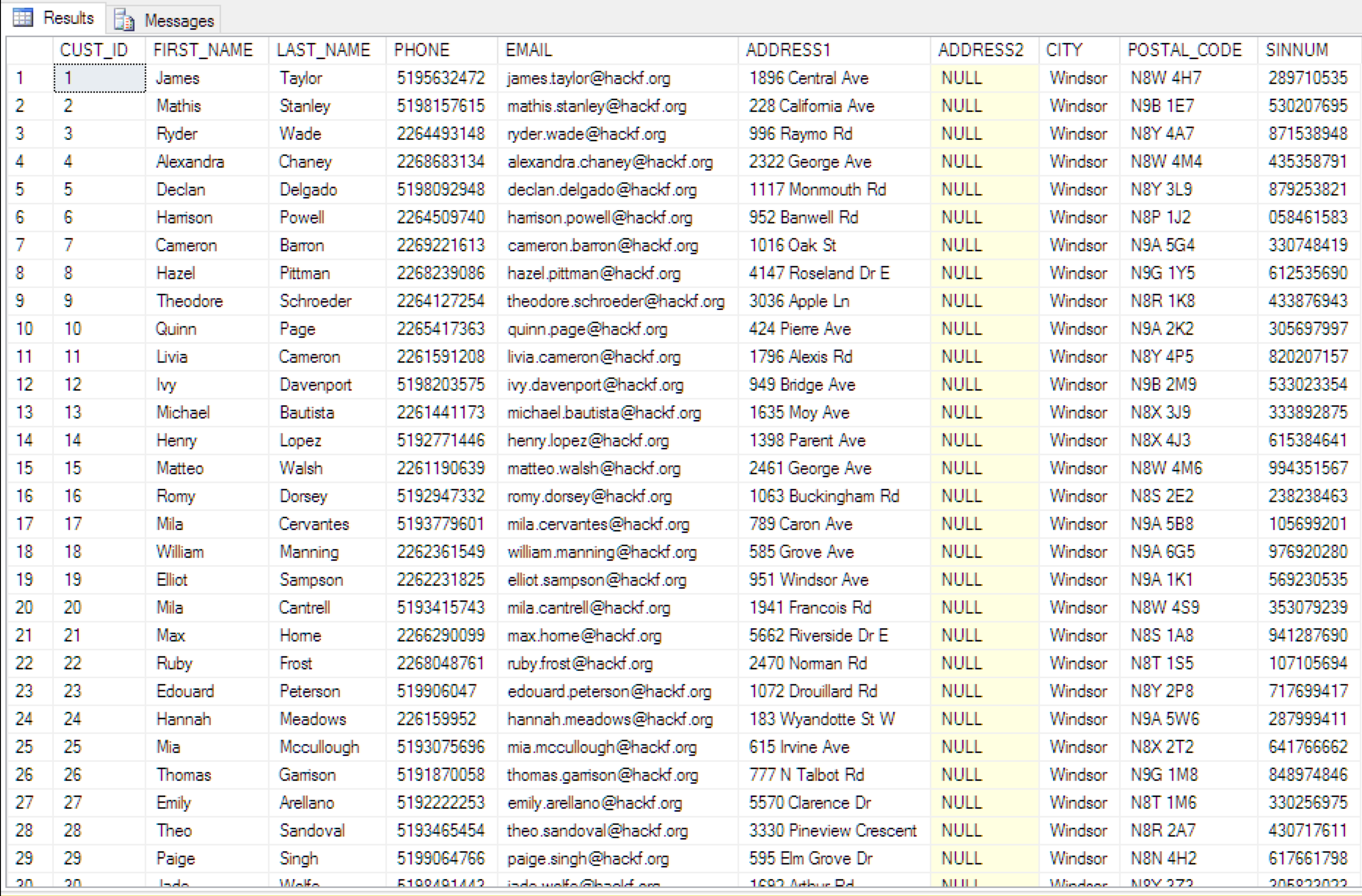
Demo

Prod Data



	CUST_ID	FIRST_NAME	LAST_NAME	PHONE	EMAIL	ADDRESS1	ADDRESS2	CITY	POSTAL_CODE	SINNUM
1	1	Lincoln	Moss	5192124699	lmoss@gmail.com	5035 Tecumseh Rd E	NULL	Windsor	N8T 1C2	880704135
2	2	Anthony	Le	5194764711	anthony.le@gmail.com	687 Sprucewood Ave	NULL	Windsor	N9C 0B3	728496779
3	3	Zoe	Santiago	2268981502	zsantiago@gmail.com	1130 Janette Ave	NULL	Windsor	N9A 5A5	142699248
4	4	Eva	English	2262543887	evaenglish@hackf.org	10463 Pulbrook Rd	NULL	Windsor	N8R 1C2	879845444
5	5	Violet	Mays	5194849035	violetmays@gmail.com	1358 Tecumseh Rd W	NULL	Windsor	N9B 1T5	570702845
6	6	Sebastian	Hancock	2260627797	shancock@uwindsor.ca	2279 Everts Ave	NULL	Windsor	N9B 3W7	154697080
7	7	Matteo	Abbott	2265178554	matteo.abbott@uwindsor.ca	2733 Everts Ave	NULL	Windsor	N9E 2T9	783195480
8	8	Hudson	Dickson	2262903780	hudson.dickson@gmail.com	3634 Byng Rd	NULL	Windsor	N8W 3H9	954905931
9	9	Simon	Kerr	2268139612	simonkerr@uwindsor.ca	464 St Paul Ave	NULL	Windsor	N8S 3L2	184321446
10	10	Aurora	Reilly	2262374864	aurora.reilly@hackf.org	1381 Elm Ave	NULL	Windsor	N8X 2B7	974208167
11	11	Nolan	Jackson	2264352491	njackson@uwindsor.ca	2633 Armstrong Ave	NULL	Windsor	N8T 2G2	308702042
12	12	Lauren	Dodson	5193075565	lauren.dodson@gmail.com	2530 Tecumseh Rd W	NULL	Windsor	N9B 3R2	106206113
13	13	Mia	Oneal	2269785294	mia.oneal@gmail.com	1082 Buckingham Rd	NULL	Windsor	N8S 2E3	163209588
14	14	Austin	Gray	5193026680	austin.gray@gmail.com	297 Barracuda St	NULL	Windsor	N8W 2B1	373875954
15	15	William	Coleman	2266002885	williamcoleman@gmail.com	685 Oak St	NULL	Windsor	N9A 5E7	196715908
16	16	Asher	Cowan	5195158884	acowan@uwindsor.ca	943 Tecumseh Rd W	NULL	Windsor	N8X 2A9	983291915
17	17	Everly	Frazier	5193418710	efrazier@hackf.org	1379 Marentette Ave	NULL	Windsor	N8X 4C9	224035584
18	18	Aria	Hartman	2269644962	aria.hartman@gmail.com	3838 Riverside Dr E	NULL	Windsor	N8Y 1B5	377600077
19	19	Payton	Stein	2262601519	paytonstein@hackf.org	732 Hildegarde St	NULL	Windsor	N8X 2Z8	165530064
20	20	Daniel	Johnston	5192357565	danieljohnston@hotmail.com	3432 Harris St	NULL	Windsor	N9C 1N5	096507850
21	21	Henry	Huang	2265246359	henry.huang@hackf.org	3369 Peter St	NULL	Windsor	N9C 1J2	430942136
22	22	Ryder	Frey	2269102272	ryderfrey@hackf.org	2423 Seminole St	NULL	Windsor	N8W 3P4	904434016
23	23	Evan	Hancock	5194328258	evan.hancock@uwindsor....	1879 Pillette Rd	NULL	Windsor	N8T 1P2	446402190
24	24	Naomi	Velez	2269153426	naomi.velez@gmail.com	1085 Felix Ave	NULL	Windsor	N9C 3L5	901801456
25	25	Brooklyn	Blake	5199511812	brooklynblake@uwindsor.ca	1614 Hall Ave	NULL	Windsor	N8X 4S1	232316851
26	26	Declan	Landry	5194716989	declan.landry@hotmail.com	1543 Ford Blvd	NULL	Windsor	N8T 2C7	572139848
27	27	Oliver	Townsend	5197459158	oliver.townsend@uwindsor...	2277 Parent Ave	NULL	Windsor	N8W 2E4	220140057
28	28	Addison	Leach	5191248102	addisonleach@gmail.com	1623 Riverside Dr E	NULL	Windsor	N8Y 1A1	087155396
29	29	Caleb	Murillo	519038240	caleb.murillo@hotmail.com	1439 St Luke Rd	NULL	Windsor	N8Y 1X3	922101977

Test Data



	CUST_ID	FIRST_NAME	LAST_NAME	PHONE	EMAIL	ADDRESS1	ADDRESS2	CITY	POSTAL_CODE	SINNUM
1	1	James	Taylor	5195632472	james.taylor@hackf.org	1896 Central Ave	NULL	Windsor	N8W 4H7	289710535
2	2	Mathis	Stanley	5198157615	mathis.stanley@hackf.org	228 California Ave	NULL	Windsor	N9B 1E7	530207695
3	3	Ryder	Wade	2264493148	ryder.wade@hackf.org	996 Raymo Rd	NULL	Windsor	N8Y 4A7	871538948
4	4	Alexandra	Chaney	2268683134	alexandra.chaney@hackf.org	2322 George Ave	NULL	Windsor	N8W 4M4	435358791
5	5	Declan	Delgado	5198092948	declan.delgado@hackf.org	1117 Monmouth Rd	NULL	Windsor	N8Y 3L9	879253821
6	6	Harrison	Powell	2264509740	harrison.powell@hackf.org	952 Banwell Rd	NULL	Windsor	N8P 1J2	058461583
7	7	Cameron	Barron	2269221613	cameron.barron@hackf.org	1016 Oak St	NULL	Windsor	N9A 5G4	330748419
8	8	Hazel	Pittman	2268239086	hazel.pittman@hackf.org	4147 Roseland Dr E	NULL	Windsor	N9G 1Y5	612535690
9	9	Theodore	Schroeder	2264127254	theodore.schroeder@hackf.org	3036 Apple Ln	NULL	Windsor	N8R 1K8	433876943
10	10	Quinn	Page	2265417363	quinn.page@hackf.org	424 Pierre Ave	NULL	Windsor	N9A 2K2	305697997
11	11	Livia	Cameron	2261591208	livia.cameron@hackf.org	1796 Alexis Rd	NULL	Windsor	N8Y 4P5	820207157
12	12	Ivy	Davenport	5198203575	ivy.davenport@hackf.org	949 Bridge Ave	NULL	Windsor	N9B 2M9	533023354
13	13	Michael	Bautista	2261441173	michael.bautista@hackf.org	1635 Moy Ave	NULL	Windsor	N8X 3J9	333892875
14	14	Henry	Lopez	5192771446	henry.lopez@hackf.org	1398 Parent Ave	NULL	Windsor	N8X 4J3	615384641
15	15	Matteo	Walsh	2261190639	matteo.walsh@hackf.org	2461 George Ave	NULL	Windsor	N8W 4M6	994351567
16	16	Romy	Dorsey	5192947332	romy.dorsey@hackf.org	1063 Buckingham Rd	NULL	Windsor	N8S 2E2	238238463
17	17	Mila	Cervantes	5193779601	mila.cervantes@hackf.org	789 Caron Ave	NULL	Windsor	N9A 5B8	105699201
18	18	William	Manning	2262361549	william.manning@hackf.org	585 Grove Ave	NULL	Windsor	N9A 6G5	976920280
19	19	Elliot	Sampson	2262231825	elliot.sampson@hackf.org	951 Windsor Ave	NULL	Windsor	N9A 1K1	569230535
20	20	Mila	Cantrell	5193415743	mila.cantrell@hackf.org	1941 Francois Rd	NULL	Windsor	N8W 4S9	353079239
21	21	Max	Horne	2266290099	max.horne@hackf.org	5662 Riverside Dr E	NULL	Windsor	N8S 1A8	941287690
22	22	Ruby	Frost	2268048761	ruby.frost@hackf.org	2470 Norman Rd	NULL	Windsor	N8T 1S5	107105694
23	23	Edouard	Peterson	519906047	edouard.peterson@hackf.org	1072 Drouillard Rd	NULL	Windsor	N8Y 2P8	717699417
24	24	Hannah	Meadows	226159952	hannah.meadows@hackf.org	183 Wyandotte St W	NULL	Windsor	N9A 5W6	287999411
25	25	Mia	Mccullough	5193075696	mia.mccullough@hackf.org	615 Irvine Ave	NULL	Windsor	N8X 2T2	641766662
26	26	Thomas	Garrison	5191870058	thomas.garrison@hackf.org	777 N Talbot Rd	NULL	Windsor	N9G 1M8	848974846
27	27	Emily	Arellano	5192222253	emily.arellano@hackf.org	5570 Clarence Dr	NULL	Windsor	N8T 1M6	330256975
28	28	Theo	Sandoval	5193465454	theo.sandoval@hackf.org	3330 Pineview Crescent	NULL	Windsor	N8R 2A7	430717611
29	29	Paige	Singh	5199064766	paige.singh@hackf.org	595 Elm Grove Dr	NULL	Windsor	N8N 4H2	617661798

Commercial Packages (in case that all sounded like a pain)

Informatica

IBM InfoSphere Optim Data Privacy

IBM InfoSphere DataStage

Oracle Data Masking and Subsetting Pack

Imperva Camouflage Data Masking

Challenges

Complex environments can introduce these problems:

1. To do it well, it requires an understanding of the data, the application, and their interactions – people with this knowledge are normally busy developing and testing
2. Not knowing where the sensitive data is (it's not always clear when you have 10's of thousands of tables across several databases)
3. Data conflicts with systems that are integrated (leads to challenge on needing to mask everything at once)
4. Changing the nature of the data (eg: data distributions) for complex tests
5. Practicalities mean that people tend to focus on identifiers but data outside of identifiers may be sensitive

It’s Like… Giving Your Data Fabricated Identities!*

- Database Security at a Glance
- Managing Risks
- Security and Test Environments
- Masking Mechanics
- Commercial Packages
- Challenges

* This is why John isn’t in marketing