Data Wrangling using SQL for data Analytics

Compiled & prepared by: Benjamin Tua | https://github.com/btua/DataWrangling_SQL

AGENDA

01

Overview

02

Practical Examples

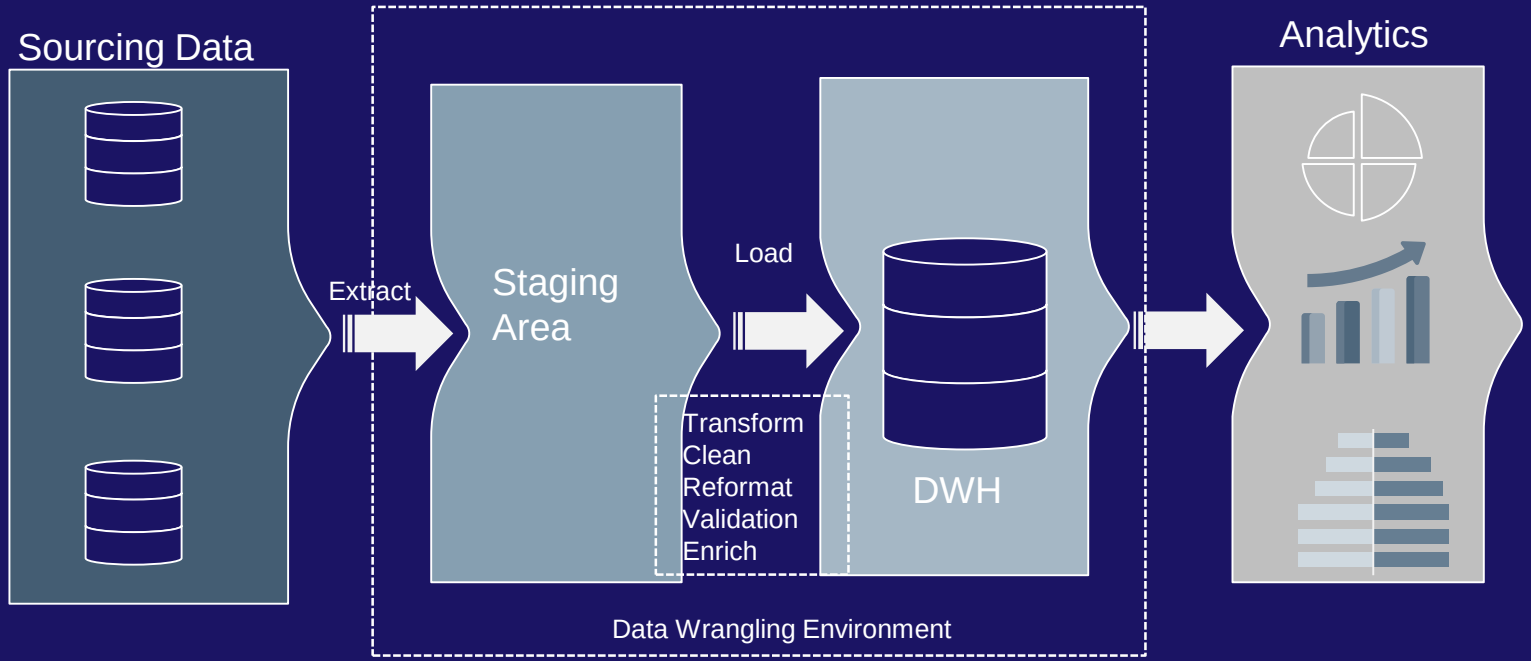
What does it mean to Wrangle data?

Data wrangling is the process of manually converting or mapping data from one "raw" form into another format that allows for more convenient consumption of the data with the help of semi-automated tools.

“Data that is loved tends to survive.”

–Kurt Bollacker

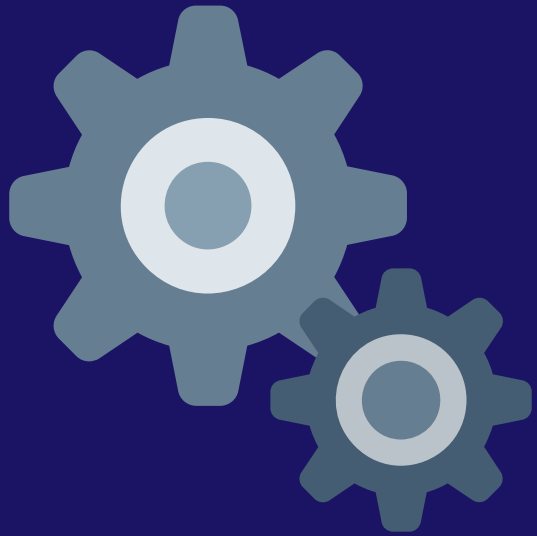
Data Process Flow



Steps in Data Wrangling

Tools of data wrangling

- CSVKit
- Python
- R
- Tabula
- Excel
- OpenRefine
- SQL



DATA WRANGLING BENEFITS

saves time

faster decision making

data usability

data preparation

improves data analytics process

removes errors

improves handling big data

data analysts can focus on analysis

Accurate actionable data

SQL (Structured Query Language)

Standardized programming language that is used to manage relational databases and perform various operations on the data in them.

SQL-compliant database server products:

- Microsoft SQL Server
- Oracle Database
- IBM Db2
- SAP HANA
- SAP Adaptive Server
- Oracle MySQL
- open source PostgreSQL

What can SQL do?

execute queries

retrieve data

insert records update records

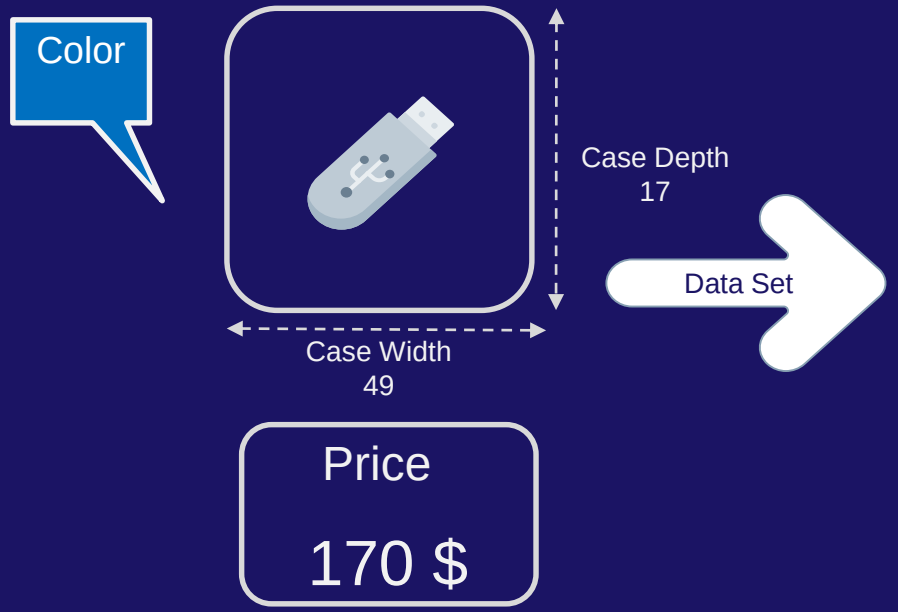
create new databases delete records

create new tables

create stored procedures

create views set permissions

Example 1



Id	Color	Case Width	Case Depth	Price
1	White	50	18	170
2	Black	49	17	200
3	Purple	46	20	NULL

Create DB

```
-- Create a new database called 'GadgetsDB'

USE master
GO
-- Create the new database if it does not exist already
IF NOT EXISTS (
    SELECT [name]
        FROM sys.databases
        WHERE [name] = N'GadgetsDB'
)
CREATE DATABASE GadgetsDB
GO
```

Create Table

```
-- Creating a FlashDrive table

CREATE TABLE FlashDrive
(
     FlashDrvId [int] IDENTITY(1,1) NOT NULL,
    Color [varchar](20) NULL,
    CaseWidth INT NULL,
    CaseDepth INT NULL,
    Price [decimal](10, 2) NULL,
    PurchaseDate Date NULL,
    CONSTRAINT [] PRIMARY KEY CLUSTERED
PK_FlashDrvType
(
    [FlashDrvId] ASC
)
)
```

Insert Data to Table

```
-- Insert rows into table 'FlashDrive‘

INSERT INTO FLASHDRIVE
( -- Columns to insert data into
 Color,CaseWidth,CaseDepth, Price, PurchaseDate
)
VALUES
(
    'Black',49,20,200.00,'2021-01-11'
),
(
    'White',45,18,150.00,'2022-02-14'
),
(
    'Purple',50,15,NULL,'2022-05-10'
),
(
    NULL,55,15,170.50,'2022-04-15'
),
(
    '-1',NULL,NULL,NULL,NULL
),
(
    '-2',NULL,20,300,'2022-03-29'
)
```

Our table output... Dataset ready for wrangling

FlashDrvId	Color	CaseWidth	CaseDepth	Price	PurchaseDate
1	Black	49	20	200.00	2021-01-11
2	White	45	18	150.00	2022-02-14
3	Purple	50	15	NULL	2022-05-10
4	NULL	55	15	170.50	2022-04-15
5	-1	NULL	NULL	NULL	NULL
6	-2	NULL	20	300.00	2022-03-29

How to find the invalid values

```
-- Get both numeric and negative Color column values

SELECT *, ISNUMERIC(Color) FROM  FLASHDRIVE
WHERE ISNUMERIC (Color)=1
  AND CAST(Color AS INT)<1
```

Results | Messages

	FlashDrvId	Color	CaseWidth	CaseDepth	Price	PurchaseDate	(No column name)
1	5	-1	NULL	NULL	NULL	2022-03-19	1
2	6	-2	NULL	20	300.00	2022-03-29	1

Replace invalid color values with Null

```
--Replace the invalid color values with NULL

UPDATE FLASHDRIVE
SET Color=NULL
WHERE ISNUMERIC(Color)=1
  AND CAST(Color AS INT)<1

--Alternative query
SELECT CASE WHEN ISNUMERIC (Color)=1 THEN NULL ELSE
Color END AS Color,
CaseWidth,CaseDepth, Price, PurchaseDate FROM
FLASHDRIVE

--Query table
SELECT * FROM FLASHDRIVE
```

Results | Messages

	FlashDrvId	Color	CaseWidth	CaseDepth	Price	PurchaseDate
1	1	Black	49	20	200.00	2021-01-11
2	2	White	45	18	150.00	2022-02-14
3	3	Purple	50	15	NULL	2022-05-10
4	4	NULL	55	15	170.50	2022-04-15
5	5	NULL	NULL	NULL	NULL	2022-03-19
6	6	NULL	NULL	20	300.00	2022-03-29

Eliminating problematic Nulls

```
-- look for all the rows where Color is Null
SELECT * FROM FLASHDRIVE
where COLOR IS NULL

-- look for all the rows where CaseWidth is Null
SELECT * FROM FLASHDRIVE
where CaseWidth IS NULL

-- look for all the rows where CaseDepth is Null
SELECT * FROM FLASHDRIVE
where CaseDepth IS NULL
```

FlashDrvId	Color	CaseWidth	CaseDepth	Price	PurchaseDate
4	NULL	55	15	170.50	2022-04-15
5	NULL	NULL	NULL	NULL	2022-03-19
6	NULL	NULL	20	300.00	2022-03-29

FlashDrvId	Color	CaseWidth	CaseDepth	Price	PurchaseDate
5	NULL	NULL	NULL	NULL	2022-03-19
6	NULL	NULL	20	300.00	2022-03-29

FlashDrvId	Color	CaseWidth	CaseDepth	Price	PurchaseDate
5	NULL	NULL	NULL	NULL	2022-03-19

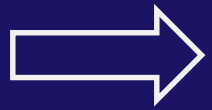
Eliminating problematic Nulls(Excessive Nulls)

```
--Eliminate problematic nulls from the FlashDrive table

DELETE FROM FLASHDRIVE
where Color is NULL
AND CaseDepth IS NULL
AND CaseWidth IS NULL
AND Price IS NULL

--Alternative query
SELECT * FROM FLASHDRIVE WHERE FlashDrvId <> 5

--View FlashDrive table after Deleting
SELECT * FROM FLASHDRIVE
```

FlashDrvId	Color	CaseWidth	CaseDepth	Price	PurchaseDate
1	Black	49	20	200.00	2021-01-11
2	White	45	18	150.00	2022-02-14
3	Purple	50	15	NULL	2022-05-10
4	NULL	55	15	170.50	2022-04-15
6	NULL	NULL	20	300.00	2022-03-29

Example 2 – Unpivoting

ITEM	2018	2019	2020	2021	2022
Flash Drive	—	—	—	—	—
Ear Phone	—	—	—	—	—
Mobile Phone	—	—	—	—	—



ITEM	YEAR	REVENUE
Flash Drive	—	—
Ear Phone	—	—
Phone	—	—
Flash Drive	—	—
Ear Phone	—	—
Phone	—	—

Create Table

```
--Create Electronic Table


CREATE TABLE [Electronic_Revenue]
(
    Item [nvarchar](50) NULL,
    [2018] [decimal](10, 2) NULL,
    [2019] [decimal](10, 2) NULL,
    [2020] [decimal](10, 2) NULL,
    [2021] [decimal](10, 2) NULL,
    [2022] [decimal](10, 2) NULL,
)
```

Insert Data to Table

```
-- Insert rows into table Electronics_Revenue

INSERT INTO [Electronics_Revenue]

(Item,[2018],[2019],[2020],[2021],[2022])

VALUES

('Flash Drive',2742.2,2892.9,3177.8,3279.3,3037.7),
('Ear Phone',1273.8,1264.1,1283.3,1199.9,1198.5),
('Mobile Phone',591,1158,2048,2982,3175),
('Mouse',1372.2,1361.2,1254.5,1233.5,1501.8),
('Printer',620.2,692.7,818,834.4,724.3)
```

Item	2018	2019	2020	2021	2022
Flash Drive	2742.20	2892.90	3177.80	3279.30	3037.70
Ear Phone	1273.80	1264.10	1283.30	1199.90	1198.50
Mobile Phone	591.00	1158.00	2048.00	2982.00	3175.00
Mouse	1372.20	1361.20	1254.50	1233.50	1501.80
Printer	620.20	692.70	818.00	834.40	724.30

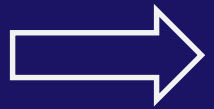
Unpivot to the desired Structure

```
-- Unpivot the table.

SELECT Item,Year,Revenue
FROM
   (SELECT Item,[2018],[2019],[2020],[2021],[2022]
      FROM Electronic_Revenue1) p
UNPIVOT
   (Revenue FOR Year IN
      ([2018],[2019],[2020],[2021],[2022])
   )AS unpvt;
GO
```

Item	Year	Revenue
Flash Drive	2018	2742.20
Flash Drive	2019	2892.90
Flash Drive	2020	3177.80
Flash Drive	2021	3279.30
Flash Drive	2022	3037.70
Ear Phone	2018	1273.80
Ear Phone	2019	1264.10
Ear Phone	2020	1283.30
Ear Phone	2021	1199.90

Example 3 – Pivoting

ITEM	YEAR	REVENUE
Flash Drive	__	__
Ear Phone	__	__
Phone	__	__
Flash Drive	__	__
Ear Phone	__	__
Phone	__	__



YEAR	Flash Drive	Ear Phone	Mobile Phone	Mouse	Printer
2018	__	__	__	__	__
2019	__	__	__	__	__
2020	__	__	__	__	__

Create & Insert Data to Table

```
--Insert the unpivoted data into a new table Electronic_Rev_Pivot

SELECT Item, Year, Revenue
into  Electronic_Rev_Pivot
FROM
   (SELECT Item, [2018],[2019],[2020],[2021],[2022]
   FROM Electronic_Revenue1) p
UNPIVOT
   (Revenue FOR Year IN
      ([2018],[2019],[2020],[2021],[2022])
)AS unpvt;
GO

Select * from Electronic_Rev_Pivot
```

Item	Year	Revenue
Flash Drive	2018	2742.20
Flash Drive	2019	2892.90
Flash Drive	2020	3177.80
Flash Drive	2021	3279.30
Flash Drive	2022	3037.70
Ear Phone	2018	1273.80
Ear Phone	2019	1264.10
Ear Phone	2020	1283.30
Ear Phone	2021	1199.90

Pivot to the desired Structure

```
-- Unpivot the table.

SELECT Year,
[Flash Drive],[Ear Phone],[Mobile Phone],[Mouse],[Printer]
FROM
(
  SELECT Item,Year,Revenue FROM Electronic_Rev_Pivot
) AS SourceTable
PIVOT
(
  AVG(Revenue)
  FOR Item IN ([Flash Drive], [Ear Phone], [Mobile
            Phone], [Mouse], [Printer])
) AS PivotTable;
```

Year	Flash Drive	Ear Phone	Mobile Phone	Mouse	Printer
2018	2742.200000	1273.800000	591.000000	1372.200000	620.200000
2019	2892.900000	1264.100000	1158.000000	1361.200000	692.700000
2020	3177.800000	1283.300000	2048.000000	1254.500000	818.000000
2021	3279.300000	1199.900000	2982.000000	1233.500000	834.400000
2022	3037.700000	1198.500000	3175.000000	1501.800000	724.300000

Example 3 - Date Formats

QUERY	FORMAT	RESULT
select convert(varchar, getdate(), 1)	mm/dd/yy	12/30/06
select convert(varchar, getdate(), 2)	yy.mm.dd	06.12.30
select convert(varchar, getdate(), 3)	dd/mm/yy	30/12/2006
select convert(varchar, getdate(), 4)	dd.mm.yy	30.12.06
select convert(varchar, getdate(), 5)	dd-mm-yy	30/12/2006
select convert(varchar, getdate(), 6)	dd-Mon-yy	30-Dec-06
select convert(varchar, getdate(), 7)	Mon dd, yy	Dec 30, 06
select convert(varchar, getdate(), 10)	mm-dd-yy	12-30-06
select convert(varchar, getdate(), 11)	yy/mm/dd	06/12/1930
select convert(varchar, getdate(), 12)	yymmdd	61230

QUERY	FORMAT	RESULT
select convert(varchar, getdate(), 23)	yyyy-mm-dd	30/12/2006
select convert(varchar, getdate(), 101)	mm/dd/yyyy	12/30/2006
select convert(varchar, getdate(), 102)	yyyy.mm.dd	2006.12.30
select convert(varchar, getdate(), 103)	dd/mm/yyyy	30/12/2006
select convert(varchar, getdate(), 104)	dd.mm.yyyy	30.12.2006
select convert(varchar, getdate(), 105)	dd-mm-yyyy	30/12/2006
select convert(varchar, getdate(), 106)	dd Mon yyyy	30-Dec-06
select convert(varchar, getdate(), 107)	Mon dd, yyyy	Dec 30, 2006
select convert(varchar, getdate(), 110)	mm-dd-yyyy	12-30-2006
select convert(varchar, getdate(), 111)	yyyy/mm/dd	30/12/2006
select convert(varchar, getdate(), 112)	yyyymmdd	20061230

Date Format Option

```
--DATE FUNCTIONS
DECLARE @counter INT = 0
DECLARE @date DATETIME = getdate()

CREATE TABLE #dateFormats (dateFormatOption int, dateOutput
nvarchar(40))

WHILE (@counter <= 150 )
BEGIN
   BEGIN TRY
      INSERT INTO #dateFormats
      SELECT CONVERT(nvarchar, @counter),CONVERT(nvarchar,@date,
@counter)
      SET @counter = @counter + 1
   END TRY
   BEGIN CATCH;
      SET @counter = @counter + 1
      IF @counter >= 150
      BEGIN
         BREAK
      END
   END CATCH
END
------
SELECT * FROM #dateFormats
------
select convert(varchar, getdate(),7)
```

Results | Messages

dateFormatOption	dateOutput
0	May 27 2022 7:05PM
1	05/27/22
2	22.05.27
3	27/05/22
4	27.05.22
5	27-05-22
6	27 May 22
7	May 27, 22
8	19:05:54
9	May 27 2022 7:05:54:860PM
10	05-27-22
11	22/05/27
12	220527
13	27 May 2022 19:05:54:860

THANK YOU!

BENJAMIN TUA