CS 513: Theory & Practice of Data Cleaning

Final Project – Phase 1 Report

University of Illinois at Urbana-Champaign, Summer 2025

Team Information

- Team ID: 112
- Members:
  - Ajithlal Parackel (ap91@illinois.edu)
  - Austin Offenberger (austino3@illinois.edu)

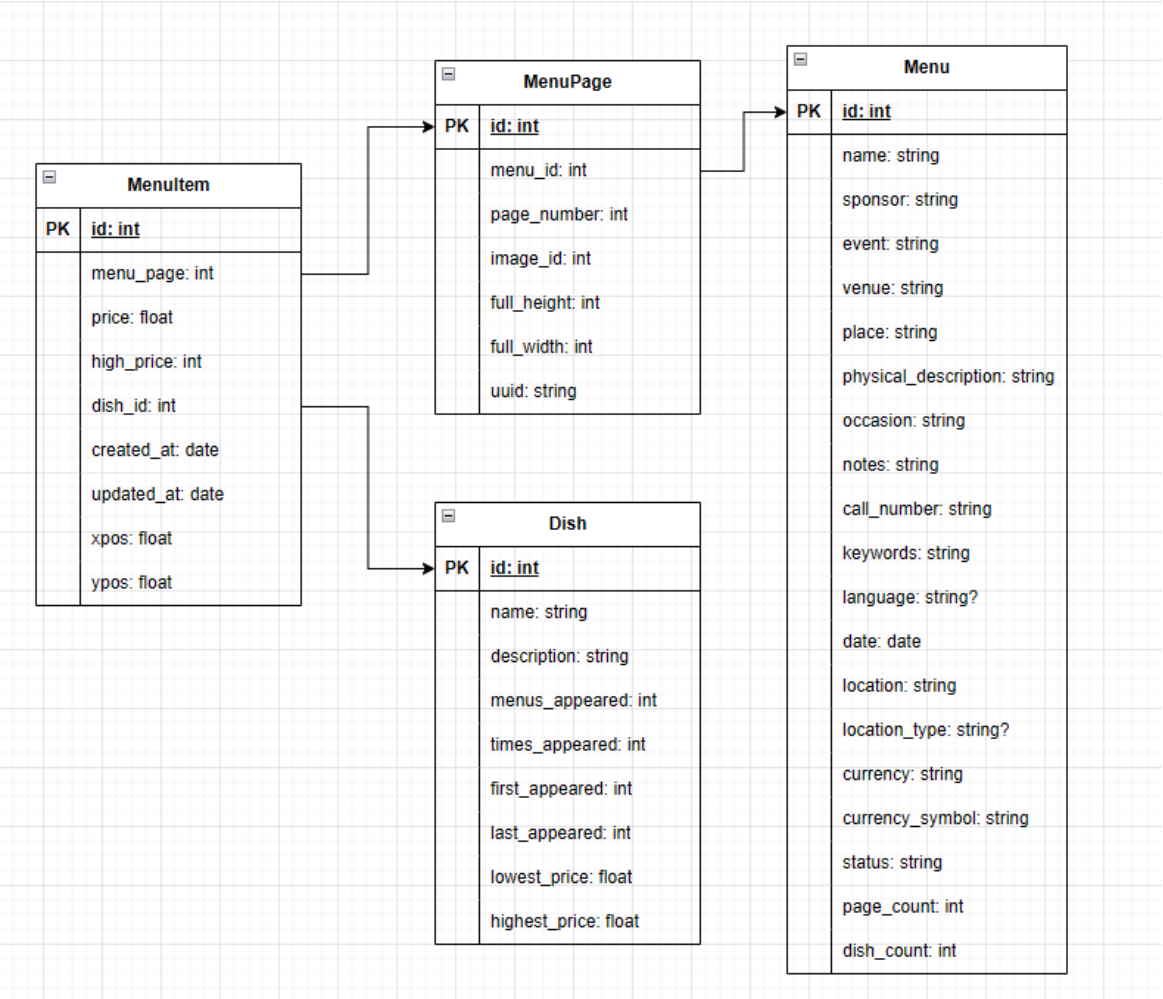
1. Description of dataset (25 points)

The NYPL Dataset extracted from digitized historical menus, which has been carefully organized into a structured format. The data is hierarchically arranged, beginning with the **menu** as the top-level unit. Each menu can contain one or more **pages**, and each page features multiple **menu items** — individual entries that typically represent **dishes** along with associated details such as prices, descriptions, etc.

In summary:

- **Menus**: Represent entire documents, usually from a single restaurant, event, or date.
- **Pages**: Account for the physical or logical divisions within a menu (e.g., front and back, or separate sections).
- **Menu Items**: Refer to the text entries on a page, each typically describing a dish or offering.
- **Dishes**: Extracted and interpreted from menu items, often requiring cleaning to resolve inconsistencies in naming, formatting, and spelling.

ER Diagram



Dish Table

Field	Description
Id	Unique string ID for each dish
Name	Name of the dish
Description	Empty
Menus Appeared	Number of menus the dish appeared in
Times Appeared	Number of times the dish appeared in menus
First Appeared	Year the dish first appeared on a menu
Last Appeared	Year the dish last appeared
Lowest Price	Lowest price listed
Highest Price	Highest price listed

Menu Page Table

Field	Description
Id	Unique string ID for each menu page
Menu ID	ID for the corresponding menu
Page Number	Page number of the menu
Image ID	ID for the image on the page
Full Height	Image height
Full Width	Image width
UUID	Second ID for either the page or the image

Menu Item Table

Field	Description
Id	Unique string ID for each menu item
Menu Page	ID of the Menu Page the item is on
Price	Price of the item
High Price	Unclear purpose
Dish ID	ID of the dish the menu item corresponds to
Created At	When the item was created
Updated At	When the item was last updated
X Pos	X position of the item on the page
Y Pos	Y position of the item on the page

Menu Table

Field	Description
Id	Unique string ID for each menu
Name	Name of the restaurant
Sponsor	Restaurant sponsor name
Event	Meal associated with the menu (e.g., Breakfast, Lunch, Dinner)
Venue	Type of menu (restaurant, private dinner, etc.)
Place	Physical location
Physical Description	Description of the actual menu
Occasion	Occasion for the menu (if any)
Notes	Continued physical descriptions
Call Number	Phone number (possibly for the restaurant)
Keywords	Empty
Language	Empty (usually the language of the menu, sometimes in Notes)
Date	Date menu was created
Location	Name of the restaurant
Location Type	Empty
Currency	Currency used
Currency Symbol	Symbol of the currency
Status	Is the menu complete?
Page Count	Number of pages in the menu
Dish Count	Number of dishes on the menu

2. Use Cases (30 points)

U1 – Target Use Case: Tracking Dish Pricing Over Time

Question 1. How has the price of coffee changed from 1900 to 2000?

Question 2. Identify the dish whose price increased the most relative to the average price change of all dishes from 1900 to 2000?

Cleaning Required:

- Normalize price formats (e.g., "1.00", "1")
- Standardize dish names (e.g., "Coffee", "Cup of Coffee", "coffee")
- Handle missing or misaligned prices

Why It's Sufficient:

- Enables time-series analysis
- Supports average price calculations per decade

U0 – Zero Data Cleaning Use Case – Popular dish.

Question 1: What are the most frequent dishes across all menus?

Note: Can be answered without cleaning, using raw frequency counts.

U2 – "Never Enough" Use Case –Price comparison and prediction.

Question 1: Was steak more expensive in NYC than in Chicago in 1920?

Question 2: Is coffee going to be expensive in 10 years?

Challenges:

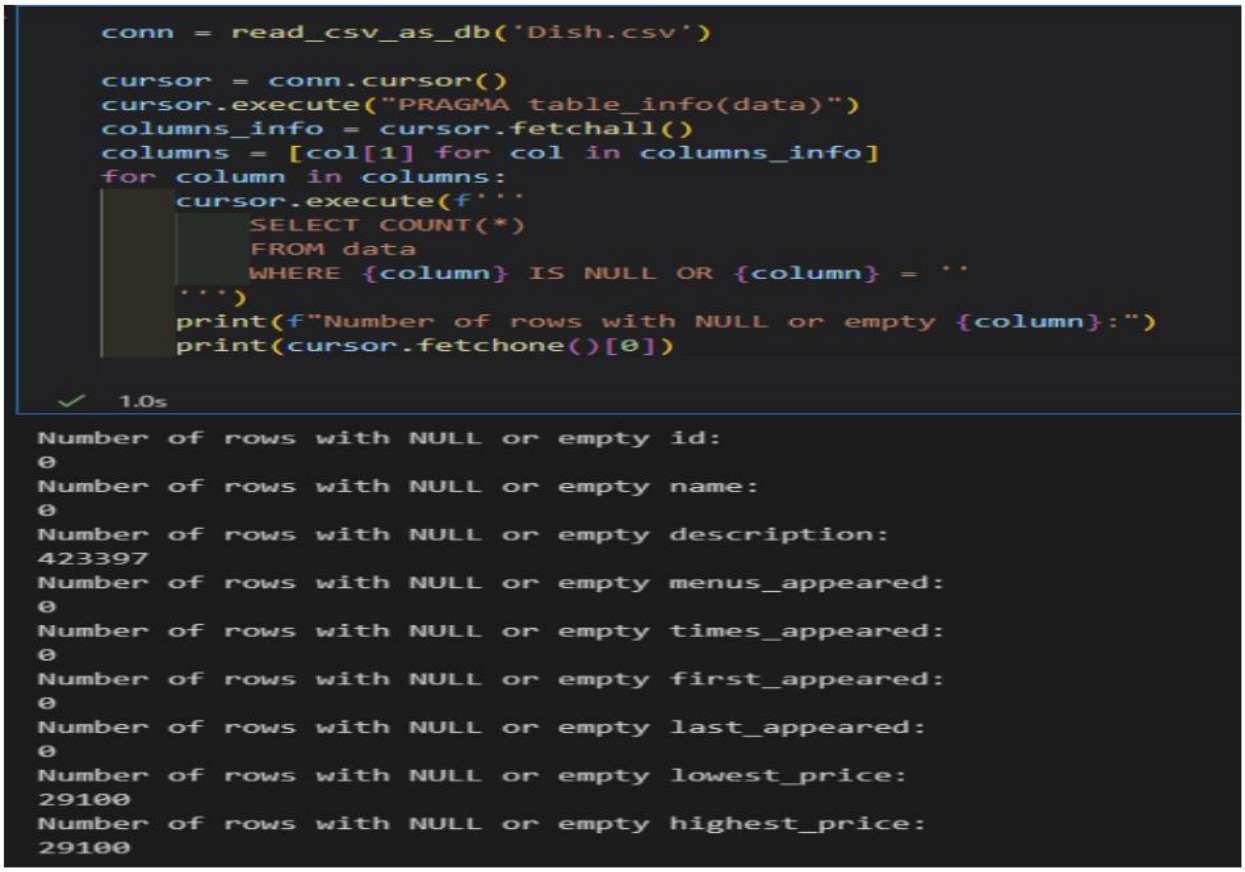
- Ambiguous units (portion size, inflation, currency)
- Price can change for each restaurant in the same location.
- Even with cleaning, normalization is unreliable

3. Data Quality Problems (30 points)

Dish Table

1.Blank/Empty price values:

Empty price values can skew or invalidate our average price calculations, so they should be handled or excluded appropriately.



```
conn = read_csv_as_db('Dish.csv')

cursor = conn.cursor()
cursor.execute("PRAGMA table_info(data)")
columns_info = cursor.fetchall()
columns = [col[1] for col in columns_info]
for column in columns:
    cursor.execute(f'''
        SELECT COUNT(*)
        FROM data
        WHERE {column} IS NULL OR {column} = ''
    ''')
    print(f"Number of rows with NULL or empty {column}:")
    print(cursor.fetchone()[0])
```

✓ 1.0s

```
Number of rows with NULL or empty id:
0
Number of rows with NULL or empty name:
0
Number of rows with NULL or empty description:
423397
Number of rows with NULL or empty menus_appeared:
0
Number of rows with NULL or empty times_appeared:
0
Number of rows with NULL or empty first_appeared:
0
Number of rows with NULL or empty last_appeared:
0
Number of rows with NULL or empty lowest_price:
29100
Number of rows with NULL or empty highest_price:
29100
```

2.No currency listed for lowest or highest price columns.

3. Dish names are not accurate.

Use Case U1 aims to identify dishes whose prices have increased the most over time compared to the average price change. However, the quality of dish name data presents significant obstacles:

- Some entries are not actual names but contain only **numbers or prices**.
- **Leading or trailing spaces** affect consistency and matching.
- Some dish names are **missing entirely**, represented by **empty quotes** ("").

7846	" " au gratin	1
131251	" " " tomatoes	1
131275	" " mushrooms	1
131285	" " mushrooms	1

id	name
486662	"
637	&
334325	,
178887	-
20332	----------
513312	... .
390496	/
25703	?
156737	??
363924	???
391736	????
404234	$1.00 DINNER
384993	$1.00 extra for each additional person
85029	$1.00 extra for Pints in Cases of 24.
262790	$125 per person including one glass each of Amontillado Sherry, Veuve Clicquot, Clos de Vougeot and …
404235	$1.50 DINNER
248205	$1.85 CHEF'S SPECIAL DINNER $1.85
420123	$3.00-Rush Luncheon- $3.00 (Choice of Entree and Coffee)
420124	$3.50- Luncheon - $3.50
157483	$4.50
157591	$4.75 PER PERSON
413965	$5.00 minimum Food charge per person.
157598	$6.00 PER PERSON
167858	002 - Chateau Haut Mazeris (Canon Fronsac)
167859	004 - Chateau Greysac (Medoc)
167861	005 - Connetable de Talbot (Saint Julien)

4. No First Appeared and Last Appeared data for some dishes.

Some dishes appear in the dataset without clear information on **when they first or last appeared**. Without a reliable time range, we can't compute a trend or slope for price change over time.

```
SELECT  count(*)
FROM "dishes"
WHERE "first_appeared" = '0' AND "last_appeared" = '0'
LIMIT 50
```

count
55278

MenuItem Table

1. Empty values for dish id, price.

Missing values for dish ID and price make it impossible to track or compare price changes over time. Without valid identifiers or price data, Use Case U1 cannot be achieved, as trends cannot be calculated or linked across records.

```
conn = read_csv_as_db('MenuItem.csv')

cursor = conn.cursor()
cursor.execute("PRAGMA table_info(data)")
columns_info = cursor.fetchall()
columns = [col[1] for col in columns_info]
for column in columns:
    cursor.execute(f'''
        SELECT COUNT(*)
        FROM data
        WHERE {column} IS NULL OR {column} = ''
    ''')
    print(f"Number of rows with NULL or empty {column}:")
    print(cursor.fetchone()[0])
```

✓ 3.4s

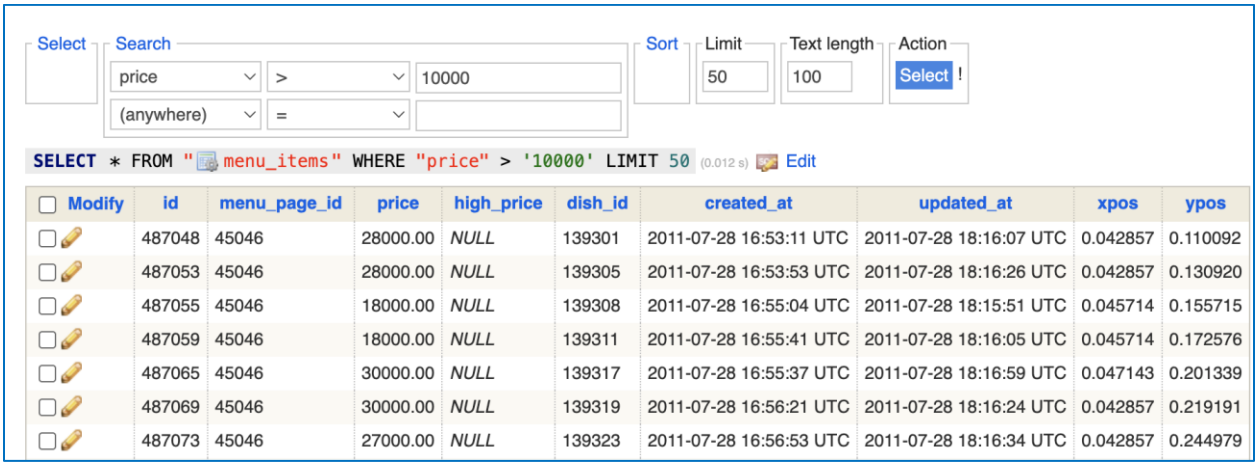
```
Number of rows with NULL or empty id:
0
Number of rows with NULL or empty menu_page_id:
0
Number of rows with NULL or empty price:
445916
Number of rows with NULL or empty high_price:
1240821
Number of rows with NULL or empty dish_id:
241
Number of rows with NULL or empty created_at:
0
Number of rows with NULL or empty updated_at:
0
Number of rows with NULL or empty xpos:
0
Number of rows with NULL or empty ypos:
0
```

2. Unclear high price column.

The **High Price** column has an unclear purpose and lacks documentation, making it difficult to interpret. Additionally, it contains zero values for some rows, which may indicate missing or invalid data, further limiting its usefulness in analysis.

3. Potentially invalid price.

Some price entries appear to be invalid, including extremely high values (e.g., >100,000) and zeros. These likely result from data entry or OCR errors and can distort trend analysis if not properly filtered or cleaned.



Modify	id	menu_page_id	price	high_price	dish_id	created_at	updated_at	xpos	ypos
	487048	45046	28000.00	*NULL*	139301	2011-07-28 16:53:11 UTC	2011-07-28 18:16:07 UTC	0.042857	0.110092
	487053	45046	28000.00	*NULL*	139305	2011-07-28 16:53:53 UTC	2011-07-28 18:16:26 UTC	0.042857	0.130920
	487055	45046	18000.00	*NULL*	139308	2011-07-28 16:55:04 UTC	2011-07-28 18:15:51 UTC	0.045714	0.155715
	487059	45046	18000.00	*NULL*	139311	2011-07-28 16:55:41 UTC	2011-07-28 18:16:05 UTC	0.045714	0.172576
	487065	45046	30000.00	*NULL*	139317	2011-07-28 16:55:37 UTC	2011-07-28 18:16:59 UTC	0.047143	0.201339
	487069	45046	30000.00	*NULL*	139319	2011-07-28 16:56:21 UTC	2011-07-28 18:16:24 UTC	0.042857	0.219191
	487073	45046	27000.00	*NULL*	139323	2011-07-28 16:56:53 UTC	2011-07-28 18:16:34 UTC	0.042857	0.244979

MenuPage Table

1. There are pages which don't have references to the menu.

Some pages in the dataset lack references to a parent menu, breaking the expected menu–page hierarchy. This makes it difficult to contextualize the dishes or associate them with a specific date, location, or restaurant.

```
select count(*) from Menu_Page MP
left outer join Menu M on M.id = MP.menu_id
where M.id is NULL
```

count
3512

2. Page number contains a null value.

It's unclear whether the record represents a valid page or a standalone single-page menu

```
conn = read_csv_as_db('MenuPage.csv')

cursor = conn.cursor()
cursor.execute("PRAGMA table_info(data)")
columns_info = cursor.fetchall()
columns = [col[1] for col in columns_info]
for column in columns:
    cursor.execute(f'''
        SELECT COUNT(*)
        FROM data
        WHERE {column} IS NULL OR {column} = ''
    ''')
    print(f"Number of rows with NULL or empty {column}:")
    print(cursor.fetchone()[0])
```

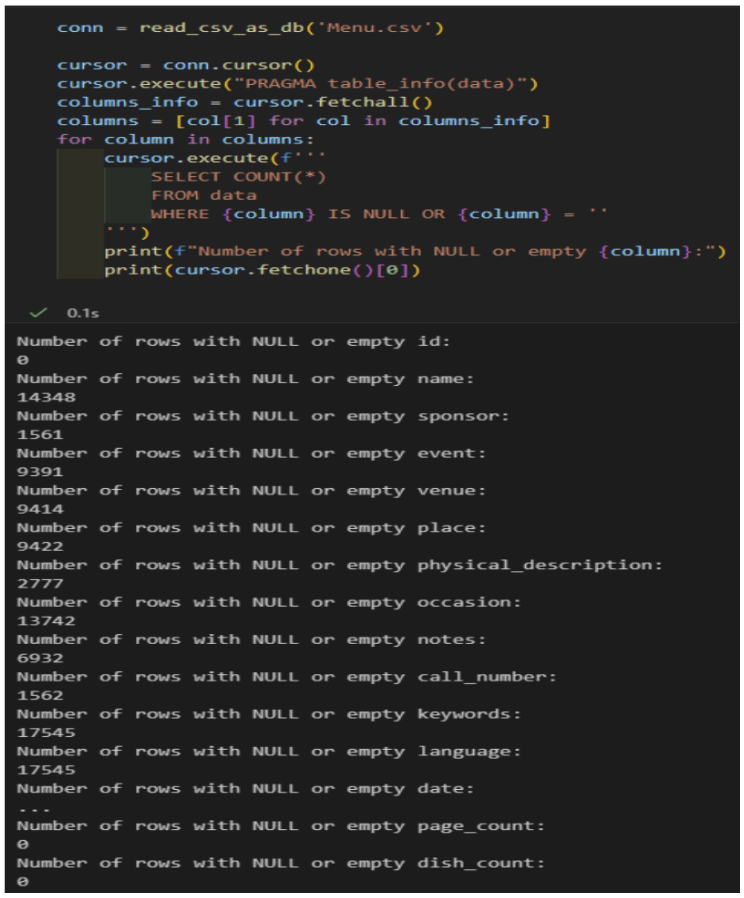
✓ 0.1s

```
Number of rows with NULL or empty id:
0
Number of rows with NULL or empty menu_id:
0
Number of rows with NULL or empty page_number:
1202
Number of rows with NULL or empty image_id:
0
Number of rows with NULL or empty full_height:
329
Number of rows with NULL or empty full_width:
329
Number of rows with NULL or empty uuid:
0
```

Menu Table

1. Empty values name, sponsor, venue, place, physical description, occasion.

Several key metadata fields — including **name**, **sponsor**, **venue**, **place**, **physical description**, and **occasion** — contain empty values in many records. These missing fields reduce the contextual richness of the data and limit its use for location-based, event-based, or provenance-related analysis.



```
conn = read_csv_as_db('Menu.csv')

cursor = conn.cursor()
cursor.execute("PRAGMA table_info(data)")
columns_info = cursor.fetchall()
columns = [col[1] for col in columns_info]
for column in columns:
    cursor.execute(f'''
        SELECT COUNT(*)
        FROM data
        WHERE {column} IS NULL OR {column} = ''
    ''')
    print(f"Number of rows with NULL or empty {column}:")
    print(cursor.fetchone()[0])
```

```
✓ 0.1s
Number of rows with NULL or empty id:
0
Number of rows with NULL or empty name:
14348
Number of rows with NULL or empty sponsor:
1561
Number of rows with NULL or empty event:
9391
Number of rows with NULL or empty venue:
9414
Number of rows with NULL or empty place:
9422
Number of rows with NULL or empty physical_description:
2777
Number of rows with NULL or empty occasion:
13742
Number of rows with NULL or empty notes:
6932
Number of rows with NULL or empty call_number:
1562
Number of rows with NULL or empty keywords:
17545
Number of rows with NULL or empty language:
17545
Number of rows with NULL or empty date:
...
Number of rows with NULL or empty page_count:
0
Number of rows with NULL or empty dish_count:
0
```

2. Currency symbol has lots of null values.

The **currency_symbol** field contains null values in many records, making it unclear which currency the listed prices refer to. This hinders accurate financial interpretation and cross-regional price comparisons.

4. Initial Plan for Phase II (15 points)

SL#	Activity	Responsibility	Timeline
S1	Review price change use case description and dataset description. - Structure and content of the dataset. - Steps to find the average price change during a time.	Ajith	7/20
S2	Profile dataset to identify data quality problems. - Basic Data Profiling - Detect Outliers - Detect Errors	Austin	7/20

	- Missing Value Analysis - Discovery of Integrity Constraint Violations		
S3	Perform data cleaning process. - Perform SQL queries to explore dataset - Use OpenRefine to perform data cleaning - Use Python scripts for further data cleaning as necessary. - Use YesWorkflow in conjunction with OpenRefine to create inner and outer workflow models Data Quality Tools: - OpenRefine - Python scripts - YesWorkflow	Ajith	7/27
S4	Identify and perform quality improvements after step 3. Find out id D is cleaner than before U1 and accurate. - Remove Errors - Exclude violations - Clustering and Facet Operations - Missing Value Operations - Exclude Outliers - Mark Integrity Constraint Violations	Austin	7/27
S5	Document and quantify change - OpenRefine Recipe - SQL Queries - Python Script - YesWorkFlow Model - Comparison of Before and After Status of the Cleaning Operation - Establish the fact that the Main Use Case U1 can be performed optimally	All	8/2