DEPI Graduation Project

Suppliers Chain Quality Project

وزارة الاتصالات
وتكنولوجيا المعلومات

CLS Learning Solutions

Our Professor:
Eng. Yasser Abdulrahman

Mohamed Selim

Youssef Ismail

Ahmed Yahia

Ahmed Hany

Ahmed Ayman

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

CLS Learning Solutions

Table of contents

01

Data Overview

Summary of tables, Metadata, and dataset structure.

02

Tech Report

Steps for data cleaning and preparation explained.

03

Analytical Report

Key trends and findings from the data analysis.

04

Take aways

Business recommendations based on the analysis.

05

Next Step

Future actions or areas for further investigation.

06

Executive Summary

Summary of insights and alignment with business goals.

01

Data Overview

About The Data

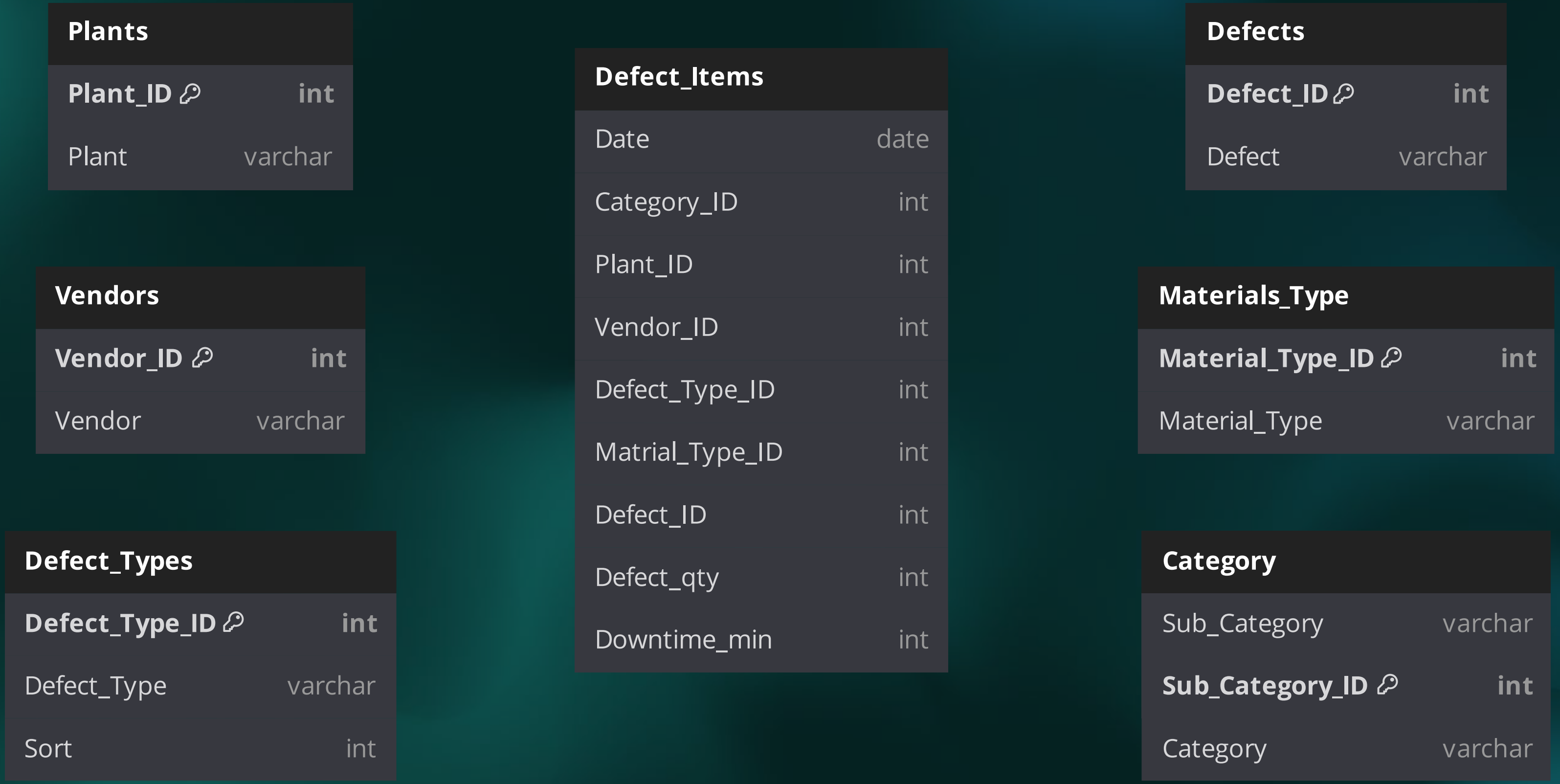
This is an analysis project of Suppliers "Vendors" Quality and Manufacturing Downtime. Here we deal with a data set of how much some vendors' materials quality is.

We aim for two main objectives:

- To understand who the best and worst suppliers are.
- Identify which plants do a better job finding and rejecting defects, to minimize downtime.

Downtime is the Time during which production is stopped, during setup for an operation, when making repairs, or when stoppages occur due to supply or labor shortages. Defects are imperfections or errors of any type that affect materials.

Entity Relationship Diagram



Plants	
Plant_ID 🔑	**int**
Plant	varchar

Vendors	
Vendor_ID 🔑	**int**
Vendor	varchar

Defect_Types	
Defect_Type_ID 🔑	**int**
Defect_Type	varchar
Sort	int

Defect_Items	
Date	date
Category_ID	int
Plant_ID	int
Vendor_ID	int
Defect_Type_ID	int
Matrial_Type_ID	int
Defect_ID	int
Defect_qty	int
Downtime_min	int

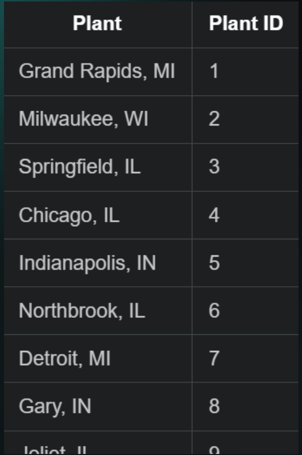
Defects	
Defect_ID 🔑	**int**
Defect	varchar

Materials_Type	
Material_Type_ID 🔑	**int**
Material_Type	varchar

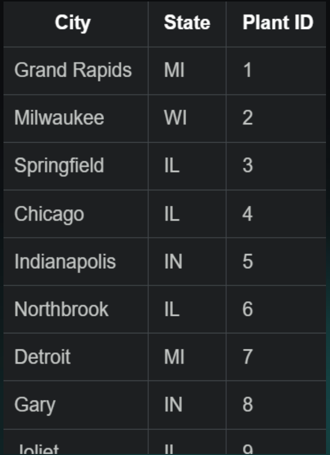
Category	
Sub_Category	varchar
Sub_Category_ID 🔑	**int**
Category	varchar

02 Technical Report

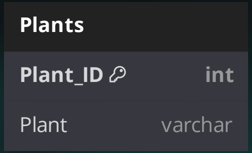
Table Preparation



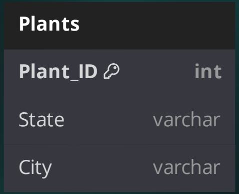
Plant	Plant ID
Grand Rapids, MI	1
Milwaukee, WI	2
Springfield, IL	3
Chicago, IL	4
Indianapolis, IN	5
Northbrook, IL	6
Detroit, MI	7
Gary, IN	8
Joliet, IL	9



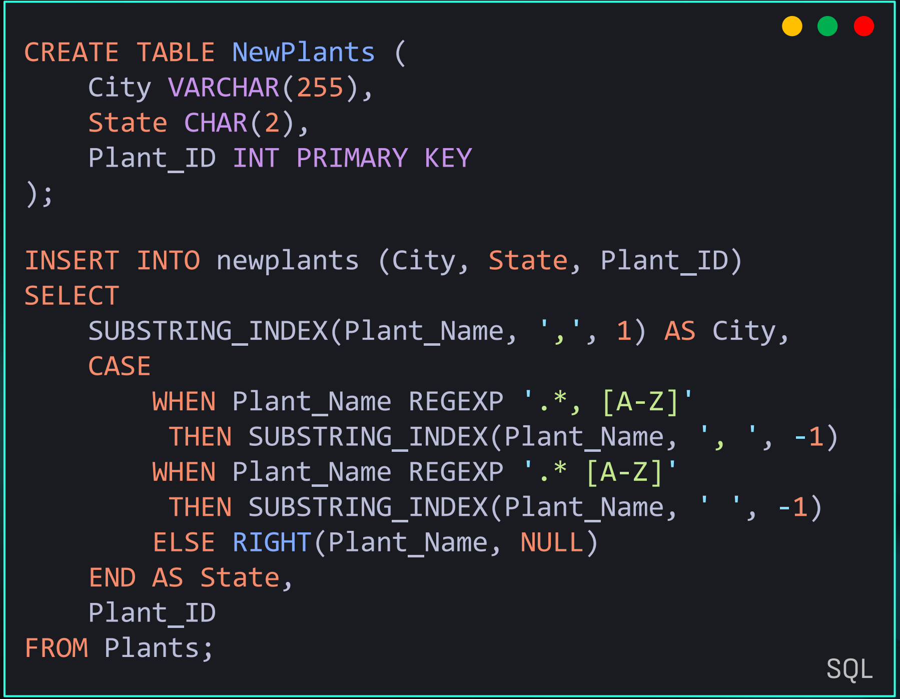
City	State	Plant ID
Grand Rapids	MI	1
Milwaukee	WI	2
Springfield	IL	3
Chicago	IL	4
Indianapolis	IN	5
Northbrook	IL	6
Detroit	MI	7
Gary	IN	8
Joliet	IL	9



Plants	
Plant_ID 🔑	int
Plant	varchar



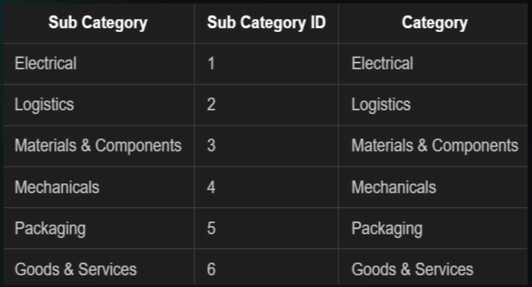
Plants	
Plant_ID 🔑	int
State	varchar
City	varchar



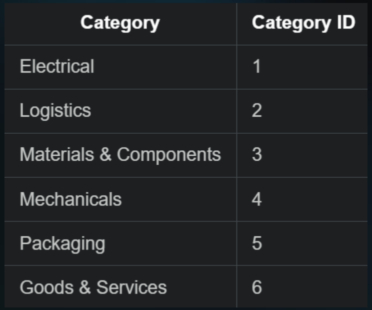
```sql
CREATE TABLE NewPlants (
    City VARCHAR(255),
    State CHAR(2),
    Plant_ID INT PRIMARY KEY
);

INSERT INTO newplants (City, State, Plant_ID)
SELECT
    SUBSTRING_INDEX(Plant_Name, ',', 1) AS City,
    CASE
        WHEN Plant_Name REGEXP '.*, [A-Z]'
         THEN SUBSTRING_INDEX(Plant_Name, ', ', -1)
        WHEN Plant_Name REGEXP '.* [A-Z]'
         THEN SUBSTRING_INDEX(Plant_Name, ' ', -1)
        ELSE RIGHT(Plant_Name, NULL)
    END AS State,
    Plant_ID
FROM Plants;
```

SQL

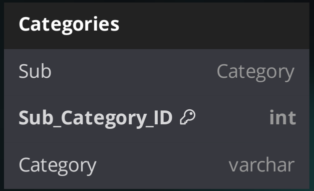
Table Preparation



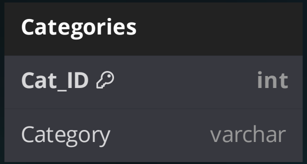
Sub Category	Sub Category ID	Category
Electrical	1	Electrical
Logistics	2	Logistics
Materials & Components	3	Materials & Components
Mechanicals	4	Mechanicals
Packaging	5	Packaging
Goods & Services	6	Goods & Services



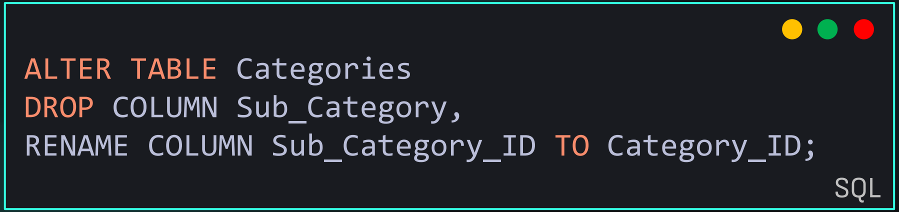
Category	Category ID
Electrical	1
Logistics	2
Materials & Components	3
Mechanicals	4
Packaging	5
Goods & Services	6



Categories	
Sub	Category
Sub_Category_ID 🔑	int
Category	varchar



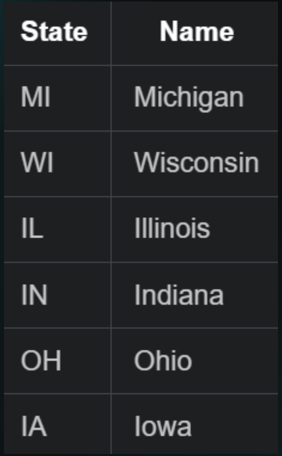
Categories	
Cat_ID 🔑	int
Category	varchar



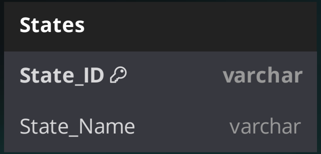
```
ALTER TABLE Categories
DROP COLUMN Sub_Category,
RENAME COLUMN Sub_Category_ID TO Category_ID;
```

SQL

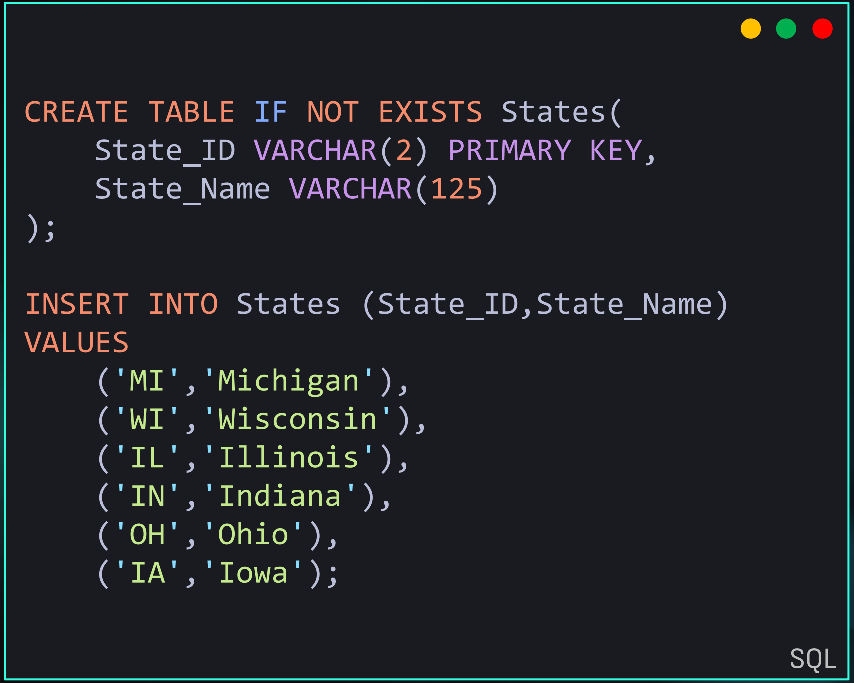
Table Preparation



State	Name
MI	Michigan
WI	Wisconsin
IL	Illinois
IN	Indiana
OH	Ohio
IA	Iowa

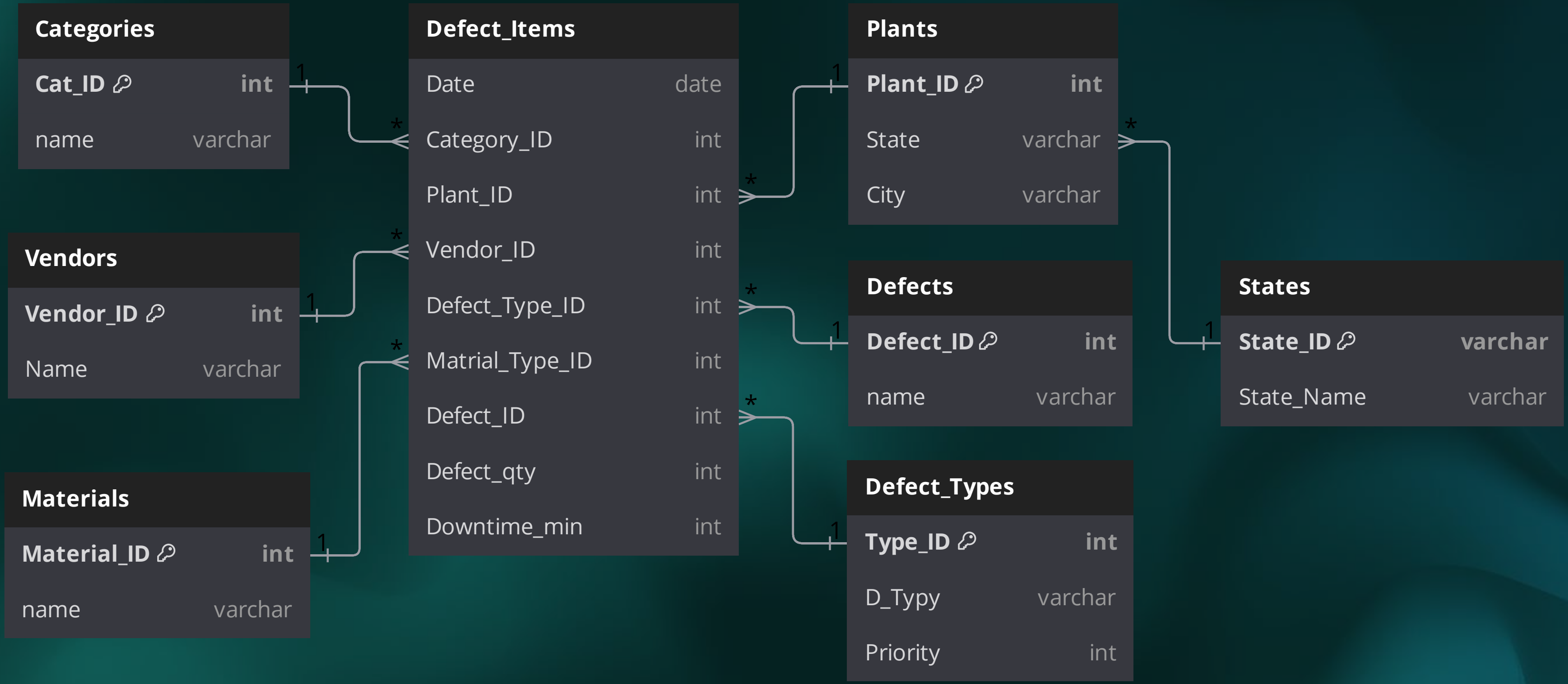


States	
State_ID 🔑	**varchar**
State_Name	varchar



```sql
CREATE TABLE IF NOT EXISTS States(
    State_ID VARCHAR(2) PRIMARY KEY,
    State_Name VARCHAR(125)
);

INSERT INTO States (State_ID,State_Name)
VALUES
    ('MI','Michigan'),
    ('WI','Wisconsin'),
    ('IL','Illinois'),
    ('IN','Indiana'),
    ('OH','Ohio'),
    ('IA','Iowa');
```

Entity Relationship Diagram



Categories	
Cat_ID 🔑	**int**
name	varchar

Defect_Items	
Date	date
Category_ID	int
Plant_ID	int
Vendor_ID	int
Defect_Type_ID	int
Matrial_Type_ID	int
Defect_ID	int
Defect_qty	int
Downtime_min	int

Plants	
Plant_ID 🔑	**int**
State	varchar
City	varchar

Vendors	
Vendor_ID 🔑	**int**
Name	varchar

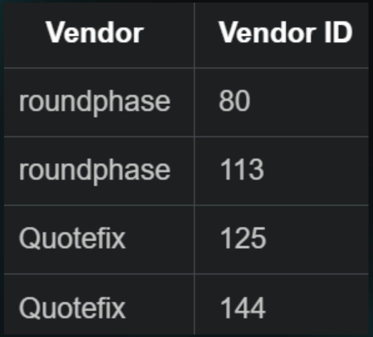
Defects	
Defect_ID 🔑	**int**
name	varchar

States	
State_ID 🔑	**varchar**
State_Name	varchar

Materials	
Material_ID 🔑	**int**
name	varchar

Defect_Types	
Type_ID 🔑	**int**
D_Typy	varchar
Priority	int

Data Cleaning



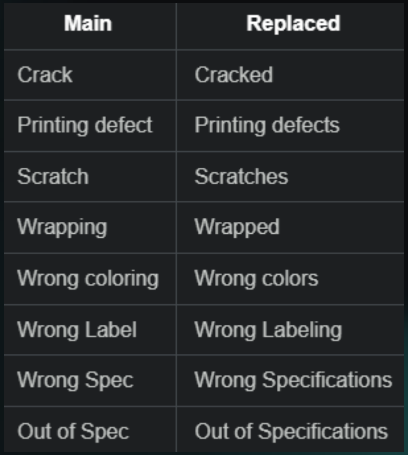
Vendor	Vendor ID
roundphase	80
roundphase	113
Quotefix	125
Quotefix	144

Remove duplicates

Data Cleaning

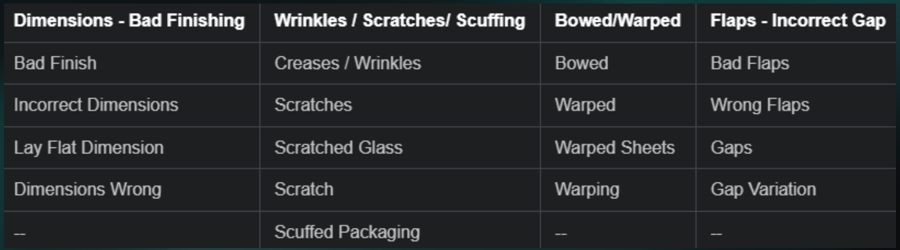
Remove duplicates [Defect Items]

Step #01



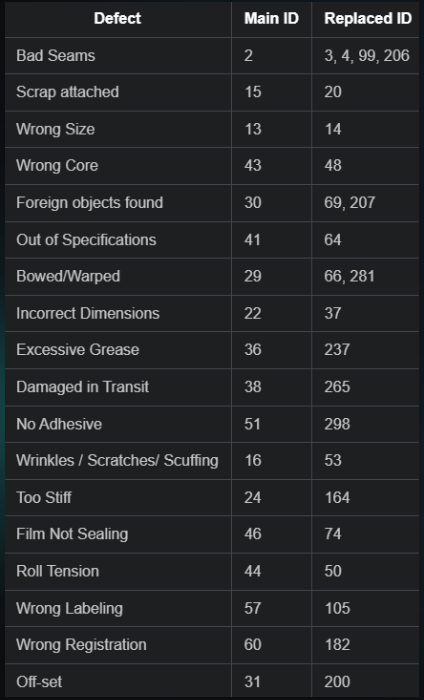
Main	Replaced
Crack	Cracked
Printing defect	Printing defects
Scratch	Scratches
Wrapping	Wrapped
Wrong coloring	Wrong colors
Wrong Label	Wrong Labeling
Wrong Spec	Wrong Specifications
Out of Spec	Out of Specifications

Step #02

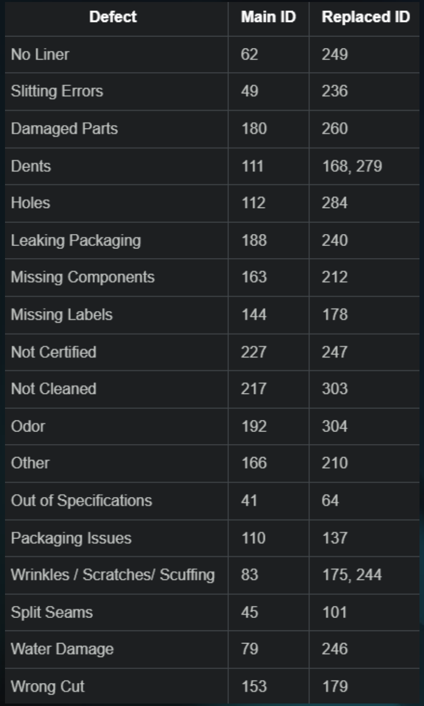


Dimensions - Bad Finishing	Wrinkles / Scratches/ Scuffing	Bowed/Warped	Flaps - Incorrect Gap
Bad Finish	Creases / Wrinkles	Bowed	Bad Flaps
Incorrect Dimensions	Scratches	Warped	Wrong Flaps
Lay Flat Dimension	Scratched Glass	Warped Sheets	Gaps
Dimensions Wrong	Scratch	Warping	Gap Variation
--	Scuffed Packaging	--	--

Step #03



Defect	Main ID	Replaced ID
Bad Seams	2	3, 4, 99, 206
Scrap attached	15	20
Wrong Size	13	14
Wrong Core	43	48
Foreign objects found	30	69, 207
Out of Specifications	41	64
Bowed/Warped	29	66, 281
Incorrect Dimensions	22	37
Excessive Grease	36	237
Damaged in Transit	38	265
No Adhesive	51	298
Wrinkles / Scratches/ Scuffing	16	53
Too Stiff	24	164
Film Not Sealing	46	74
Roll Tension	44	50
Wrong Labeling	57	105
Wrong Registration	60	182
Off-set	31	200

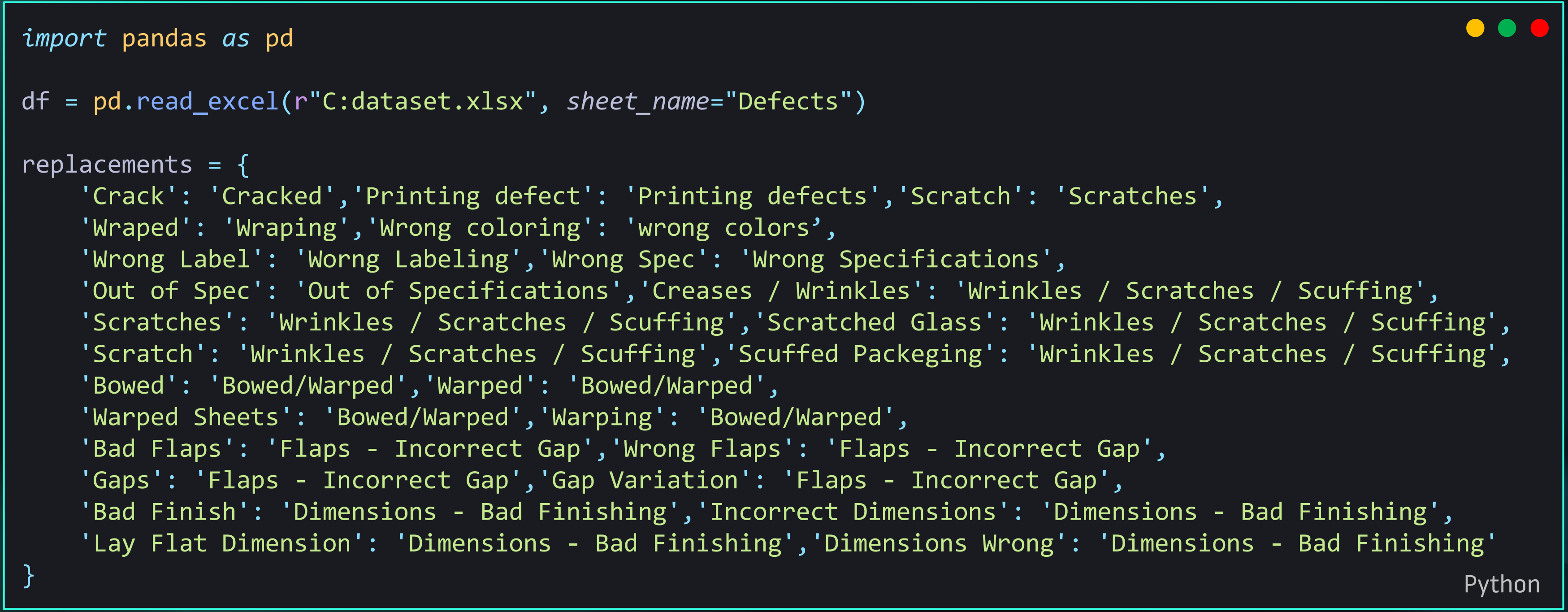


Defect	Main ID	Replaced ID
No Liner	62	249
Slitting Errors	49	236
Damaged Parts	180	260
Dents	111	168, 279
Holes	112	284
Leaking Packaging	188	240
Missing Components	163	212
Missing Labels	144	178
Not Certified	227	247
Not Cleaned	217	303
Odor	192	304
Other	166	210
Out of Specifications	41	64
Packaging Issues	110	137
Wrinkles / Scratches/ Scuffing	83	175, 244
Split Seams	45	101
Water Damage	79	246
Wrong Cut	153	179

Data Cleaning

Remove duplicates [Defect Items]

Step #01 & Step #02

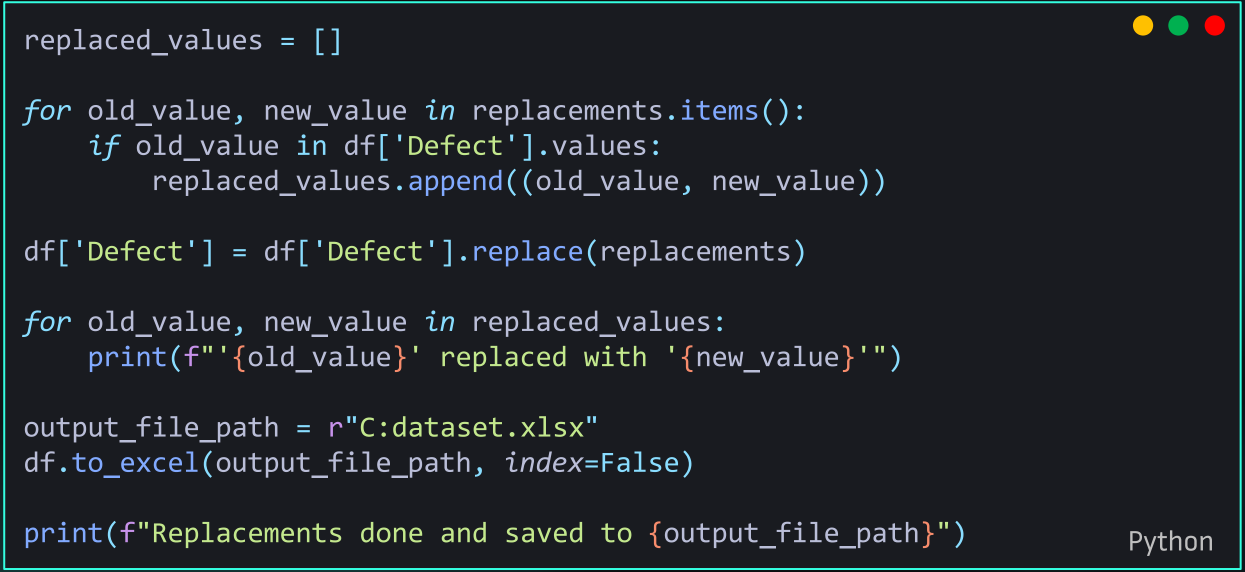


```python
import pandas as pd

df = pd.read_excel(r"C:dataset.xlsx", sheet_name="Defects")

replacements = {
    'Crack': 'Cracked','Printing defect': 'Printing defects','Scratch': 'Scratches',
    'Wraped': 'Wraping','Wrong coloring': 'wrong colors’,
    'Wrong Label': 'Worng Labeling','Wrong Spec': 'Wrong Specifications',
    'Out of Spec': 'Out of Specifications','Creases / Wrinkles': 'Wrinkles / Scratches / Scuffing',
    'Scratches': 'Wrinkles / Scratches / Scuffing','Scratched Glass': 'Wrinkles / Scratches / Scuffing',
    'Scratch': 'Wrinkles / Scratches / Scuffing','Scuffed Packeging': 'Wrinkles / Scratches / Scuffing',
    'Bowed': 'Bowed/Warped','Warped': 'Bowed/Warped',
    'Warped Sheets': 'Bowed/Warped','Warping': 'Bowed/Warped',
    'Bad Flaps': 'Flaps - Incorrect Gap','Wrong Flaps': 'Flaps - Incorrect Gap',
    'Gaps': 'Flaps - Incorrect Gap','Gap Variation': 'Flaps - Incorrect Gap',
    'Bad Finish': 'Dimensions - Bad Finishing','Incorrect Dimensions': 'Dimensions - Bad Finishing',
    'Lay Flat Dimension': 'Dimensions - Bad Finishing','Dimensions Wrong': 'Dimensions - Bad Finishing'
}
```

Data Cleaning

Step #01 & Step #02

Remove duplicates [Defect Items]

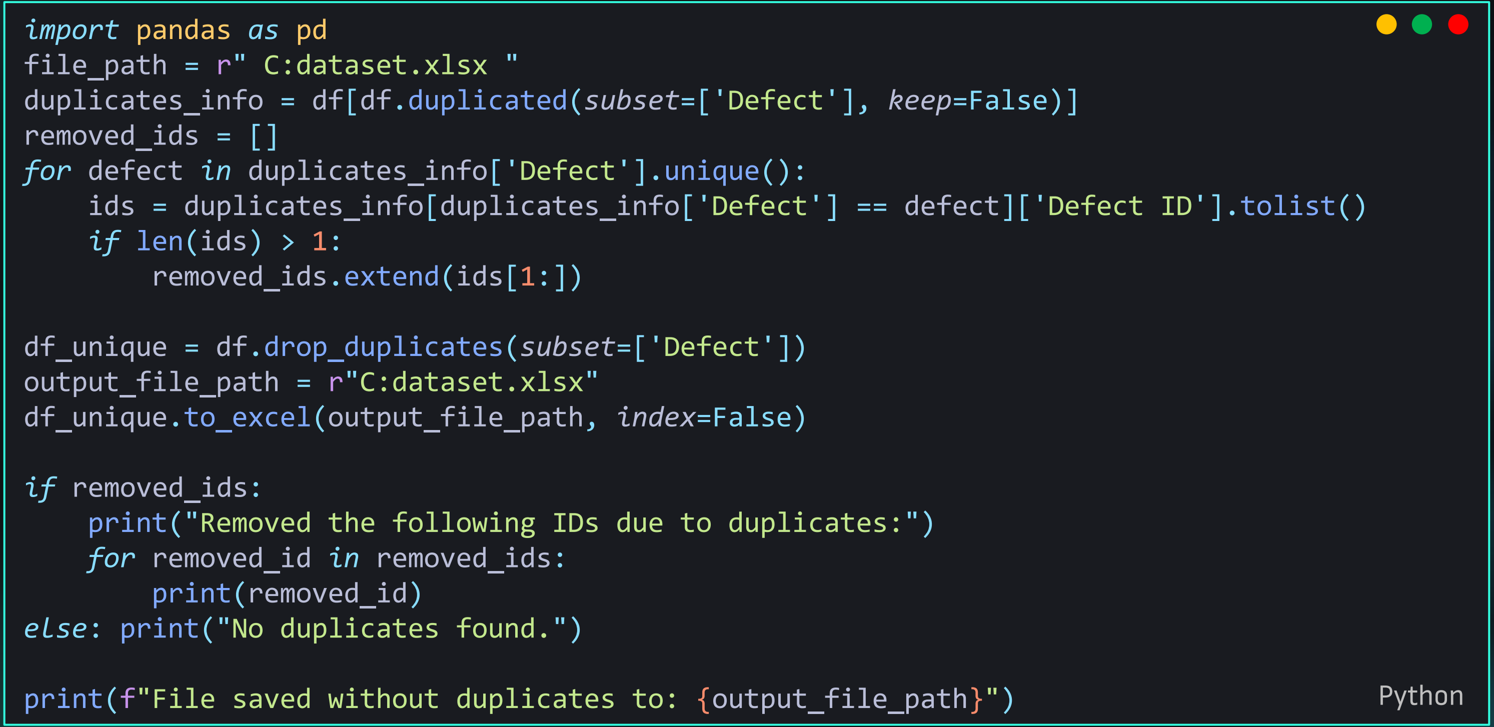


```python
replaced_values = []

for old_value, new_value in replacements.items():
    if old_value in df['Defect'].values:
        replaced_values.append((old_value, new_value))

df['Defect'] = df['Defect'].replace(replacements)

for old_value, new_value in replaced_values:
    print(f"'{old_value}' replaced with '{new_value}'")

output_file_path = r"C:dataset.xlsx"
df.to_excel(output_file_path, index=False)

print(f"Replacements done and saved to {output_file_path}")
```

Data Cleaning

Step #03

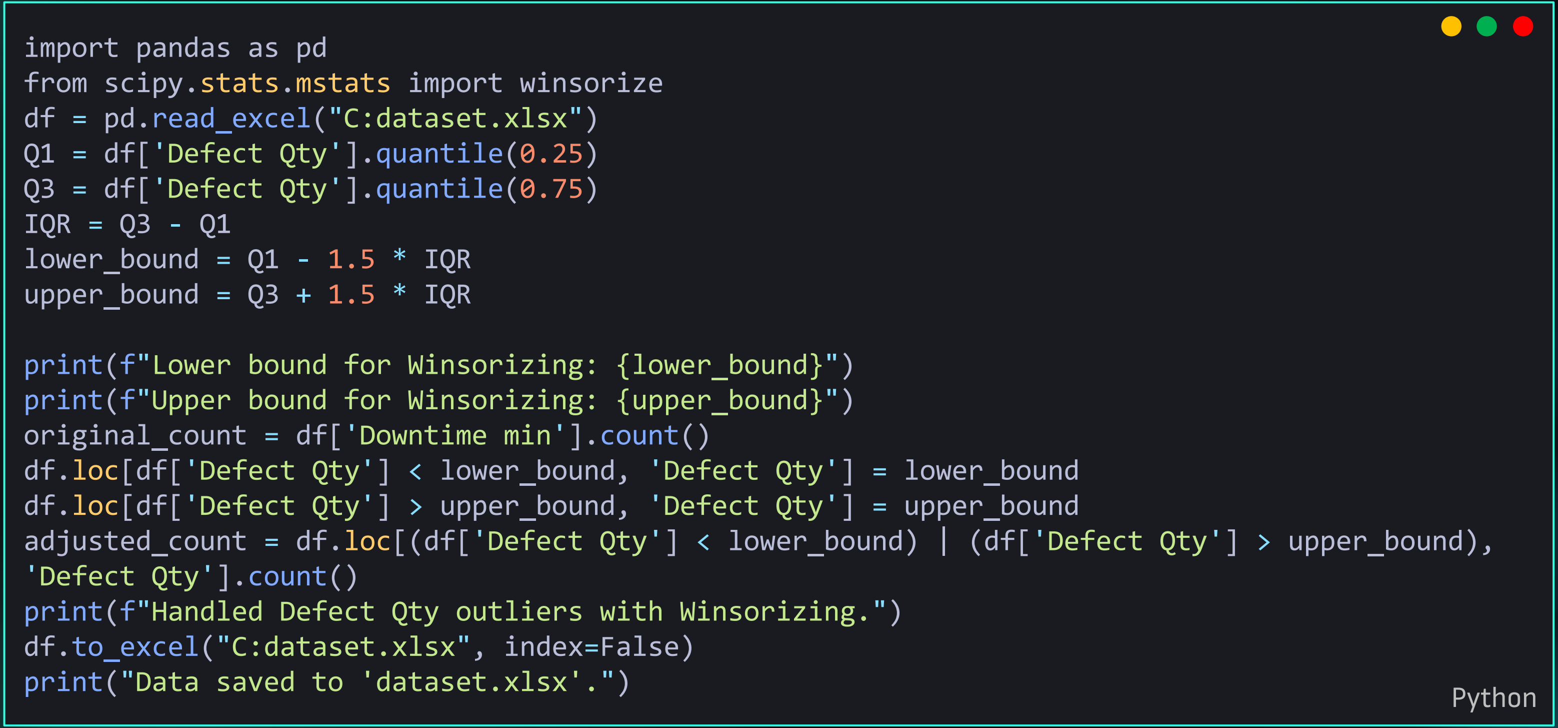
Remove duplicates [Defect Items]



```python
import pandas as pd
file_path = r" C:dataset.xlsx "
duplicates_info = df[df.duplicated(subset=['Defect'], keep=False)]
removed_ids = []
for defect in duplicates_info['Defect'].unique():
    ids = duplicates_info[duplicates_info['Defect'] == defect]['Defect ID'].tolist()
    if len(ids) > 1:
        removed_ids.extend(ids[1:])

df_unique = df.drop_duplicates(subset=['Defect'])
output_file_path = r"C:dataset.xlsx"
df_unique.to_excel(output_file_path, index=False)

if removed_ids:
    print("Removed the following IDs due to duplicates:")
    for removed_id in removed_ids:
        print(removed_id)
else: print("No duplicates found.")

print(f"File saved without duplicates to: {output_file_path}")
```

Dealing with outliers

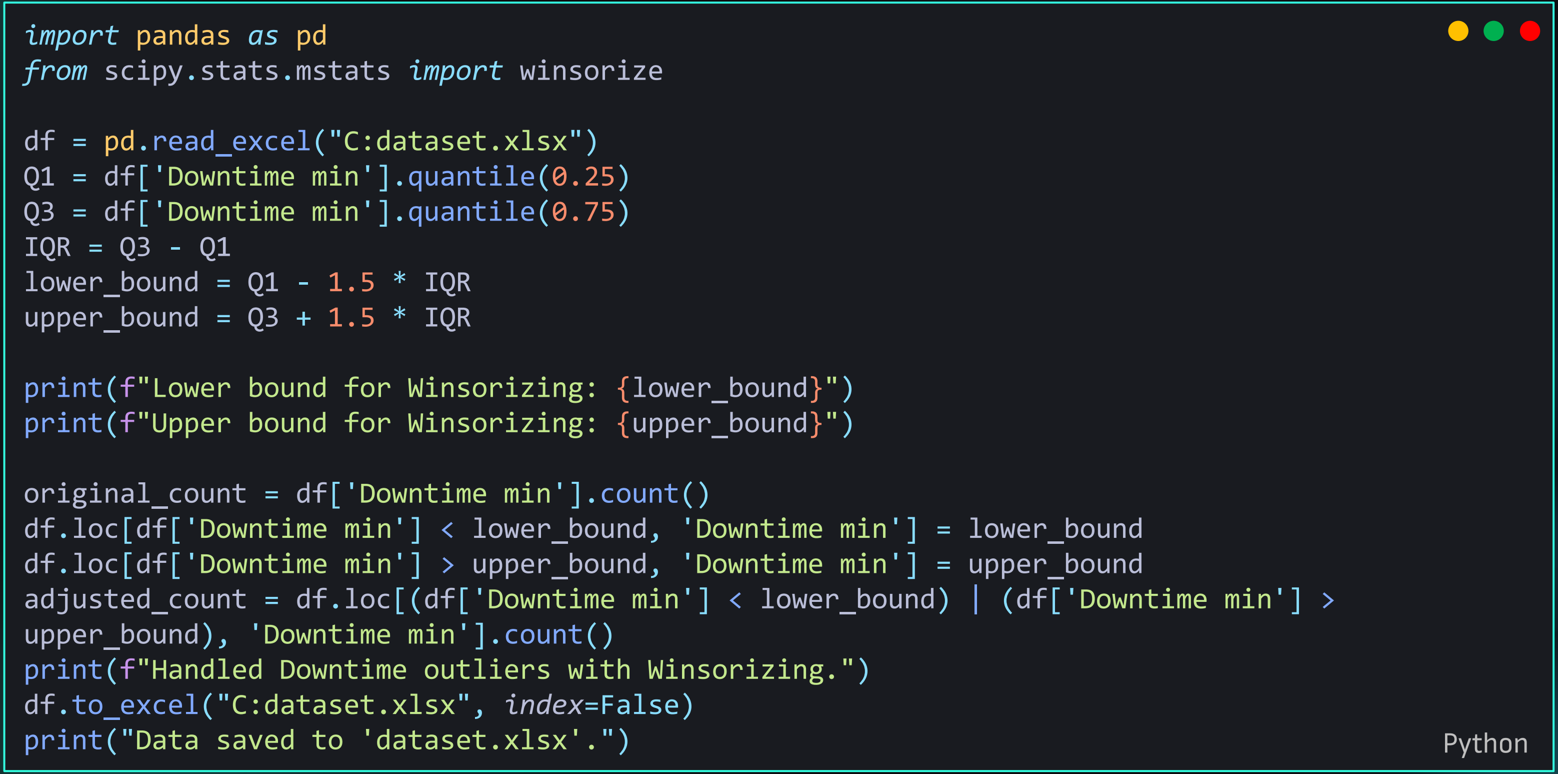
Defect quantity outliers



```python
import pandas as pd
from scipy.stats.mstats import winsorize
df = pd.read_excel("C:dataset.xlsx")
Q1 = df['Defect Qty'].quantile(0.25)
Q3 = df['Defect Qty'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"Lower bound for Winsorizing: {lower_bound}")
print(f"Upper bound for Winsorizing: {upper_bound}")
original_count = df['Downtime min'].count()
df.loc[df['Defect Qty'] < lower_bound, 'Defect Qty'] = lower_bound
df.loc[df['Defect Qty'] > upper_bound, 'Defect Qty'] = upper_bound
adjusted_count = df.loc[(df['Defect Qty'] < lower_bound) | (df['Defect Qty'] > upper_bound),
'Defect Qty'].count()
print(f"Handled Defect Qty outliers with Winsorizing.")
df.to_excel("C:dataset.xlsx", index=False)
print("Data saved to 'dataset.xlsx'.")
```

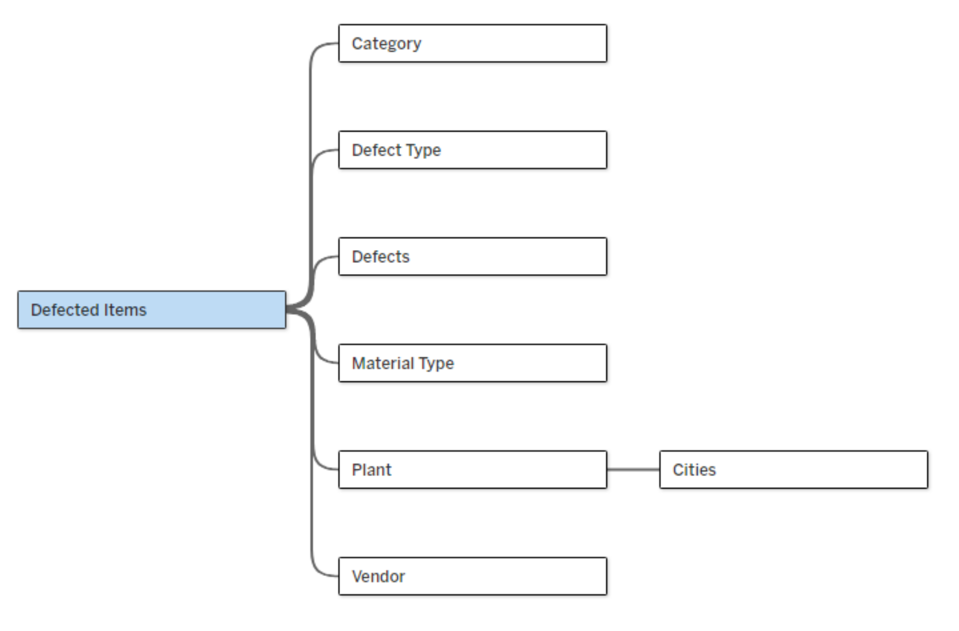
Dealing with outliers

Down time outliers



```
import pandas as pd
from scipy.stats.mstats import winsorize

df = pd.read_excel("C:dataset.xlsx")
Q1 = df['Downtime min'].quantile(0.25)
Q3 = df['Downtime min'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

print(f"Lower bound for Winsorizing: {lower_bound}")
print(f"Upper bound for Winsorizing: {upper_bound}")

original_count = df['Downtime min'].count()
df.loc[df['Downtime min'] < lower_bound, 'Downtime min'] = lower_bound
df.loc[df['Downtime min'] > upper_bound, 'Downtime min'] = upper_bound
adjusted_count = df.loc[(df['Downtime min'] < lower_bound) | (df['Downtime min'] >
upper_bound), 'Downtime min'].count()
print(f"Handled Downtime outliers with Winsorizing.")
df.to_excel("C:dataset.xlsx", index=False)
print("Data saved to 'dataset.xlsx'.")
```

Data Modeling



Defected Items

Category

Defect Type

Defects

Material Type

Plant

Cities

Vendor

03

Analytical Report

5758 Inspections

Number of Inspections

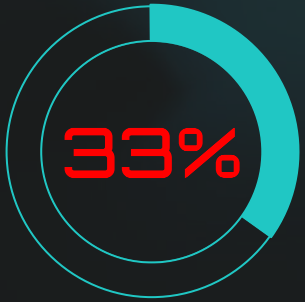
11.66 mins

AVG down time in Mins

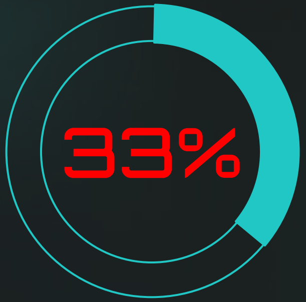
3225 defect

AVG Defects Quantity

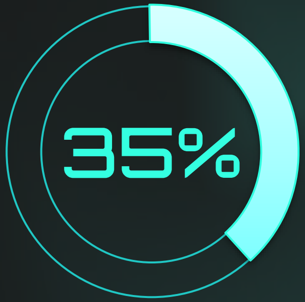
Techniques usability



Rejected

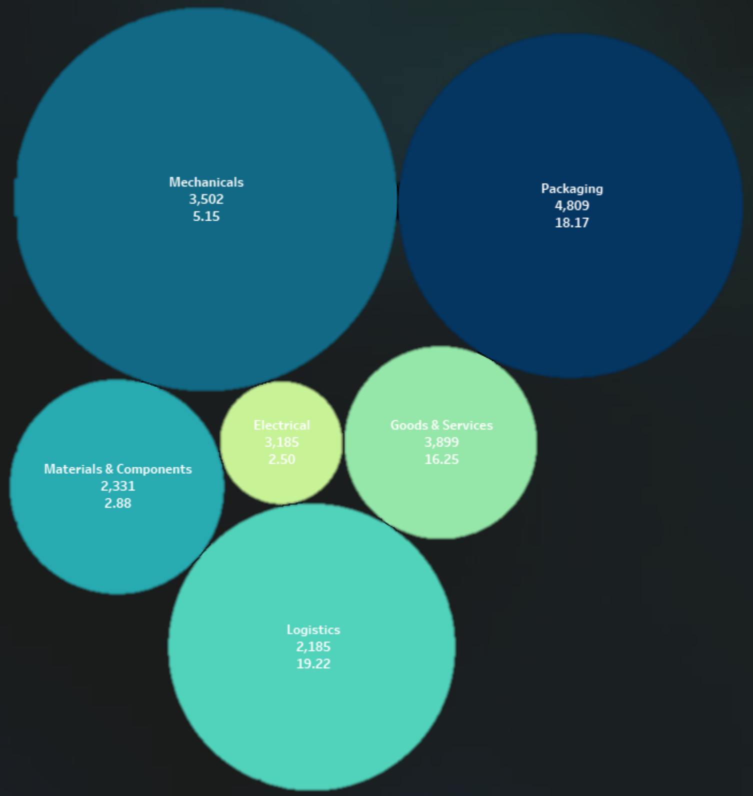


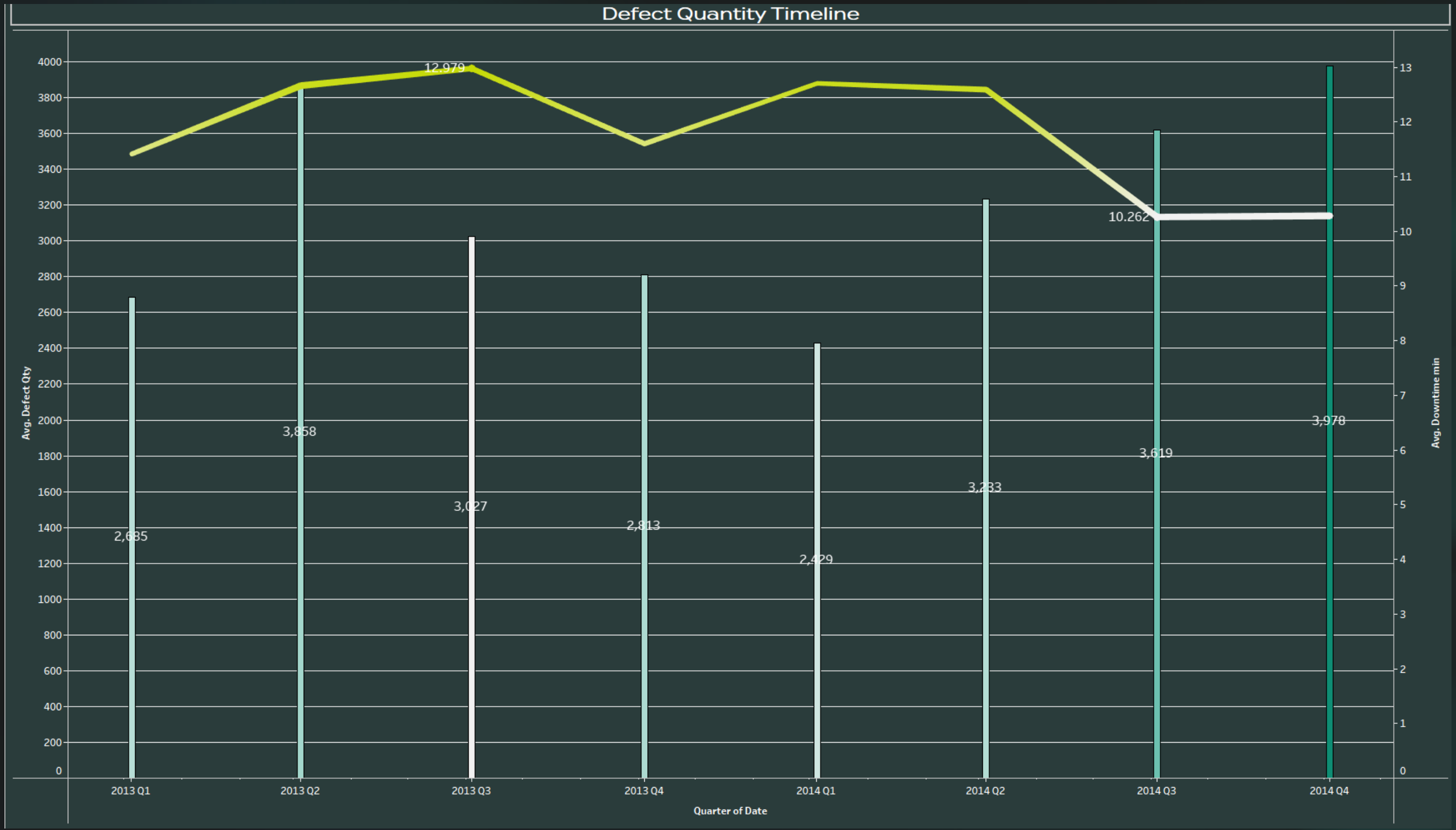
Impact



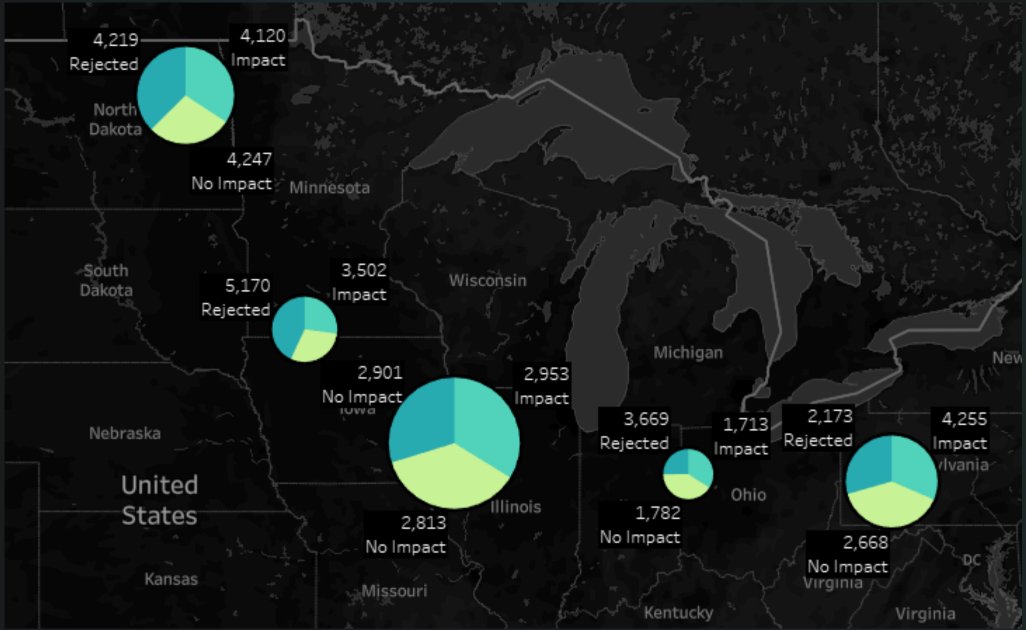
No Impact

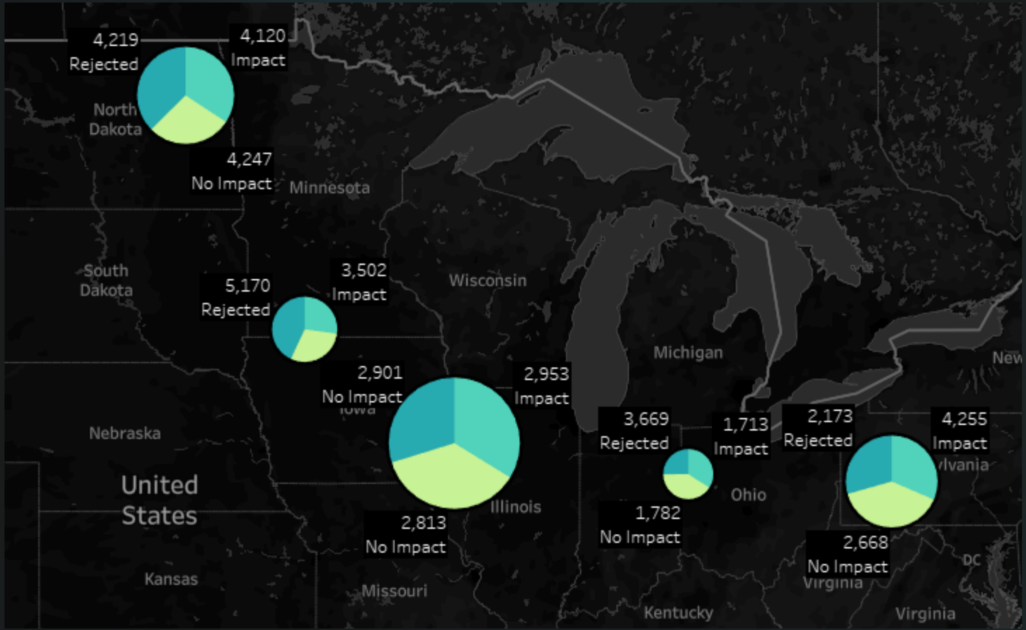
Rejected and Impact product has more than 65% of the total defects


Mechanicals
3,502
5.15
Packaging
4,809
18.17
Materials & Components
2,331
2.88
Electrical
3,185
2.50
Goods & Services
3,899
16.25
Logistics
2,185
19.22



Defect Quantity Timeline


4,219
Rejected
4,120
Impact
North
Dakota
4,247
No Impact
Minnesota
South
Dakota
5,170
Rejected
3,502
Impact
Wisconsin
2,901
No Impact
Iowa
2,953
Impact
Nebraska
United
States
2,813
No Impact
Illinois
Kansas
Missouri
Michigan
3,669
Rejected
1,713
Impact
1,782
No Impact
Ohio
2,173
Rejected
4,255
Impact
Pennsylvania
2,668
No Impact
Virginia
DC
Kentucky
Virginia
New


4,219
Rejected
4,120
Impact
North
Dakota
4,247
No Impact
Minnesota
South
Dakota
5,170
Rejected
3,502
Impact
Wisconsin
2,901
No Impact
Iowa
2,953
Impact
Michigan
New
Nebraska
United
States
2,813
No Impact
Illinois
3,669
Rejected
1,713
Impact
2,173
Rejected
4,255
Impact
ylvania
Ohio
1,782
No Impact
2,668
No Impact
Virginia
DC
Kansas
Missouri
Kentucky
Virginia