



Engenharia de Dados para Suporte à Tomada de Decisão



Joana Rodrigues Pereira Couto

A95480



Miguel Guilherme Capela Esteves Gonçalves

A96771



Inês Covas da Costa

A96986



Miguel Ferreira

A96146



Bruna Matilde Silva Fernandes

A97193

Universidade do Minho - Ano Letivo 2022/2023

Licenciatura em Engenharia de Gestão e Sistemas de Informação

Índice

Introdução	3
Identificação do Tema	4
Análise da Qualidade de Dados	5
Drinking Fountains	5
Drinking Water Quality Distribution Monitoring	13
DEP Green Infrastructure	18
Water Tank Quality	34
Harbor Water Quality	40
Questões Analíticas	46
KPI's	48
Arquitetura	49
Camada Bronze	49
Camada Silver	51
Camada Gold	68
Diagrama Pipeline	68
Conclusão	69

Introdução

Este projeto é desenvolvido no âmbito da unidade curricular de Engenharia de Dados para Apoio à Tomada de Decisão no terceiro ano de Engenharia e Gestão de Sistemas de Informação.

O projeto inclui um processo de extração, análise, modelagem e processamento de dados com o objetivo de interpretar uma grande quantidade de dados sobre um tema específico selecionado pelo grupo. O tema escolhido pelo nosso grupo é a qualidade da água.

A água é o mais crítico e importante elemento para a vida humana, e apesar de renovável deve ser estudada assim como a sua qualidade e formas de melhorá-la. Este estudo é baseado em dados de New York.

Este relatório vai apresentar cinco datasets inseridos neste tema, sendo eles:

- Localização de fontes
- Qualidade da água na distribuição
- Projetos para a melhoria da água
- Inspeção e qualidade da água dos tanques
- Qualidade da água nos portos

Numa perspetiva de aprendizagem, as equipas devem ser capazes de consolidar e aplicar conhecimentos relacionados com outras unidades curriculares, bem como desenvolver novas competências.

Identificação do Tema

O tema escolhido para o desenvolvimento deste projeto foi a qualidade da água.

Porquê?

- A água pode ser considerada o nutriente mais importante da vida dos seres humanos;
- Equilibra e conserva a biodiversidade;
- O fornecimento de água limpa e fresca é fundamental para a saúde;
- A importância do tratamento de águas residenciais e industriais está diretamente atrelada a escassez de água potável;
- A análise de portos possibilita testar os efeitos de várias práticas de manejo no uso da terra, ou os efeitos de poluentes ambientais em sistemas naturais;
- A água é essencial para o consumo, por exemplo, para limpeza, higiene e preparação de alimentos;

Objetivos :

- Perceber os indicadores que poluem a água;
- Perceber os indicadores que ajudam na melhoria da água;
- Entender de que forma as iniciativas de projetos DEP Green ajudam na melhoria da qualidade de água;
- Interpretar a qualidade da água na distribuição;
- Compreender a influência da qualidade da água dos portos;

Questões Gerais

- Quais as localizações das fontes provenientes da água dos tanques?
- Quais os projetos DEP Green apresentam menor valor de turbidity associadas aos portos?
- Tendo como indicador a presença de E.coli, em que ano (a partir de 2018) é que houve uma melhor qualidade da água das fontes?
- Quais são os parques que apresentam melhor qualidade de água?
- Existe uma melhoria da qualidade de água devido aos projetos DEP Green?

Análise da Qualidade de Dados

Drinking Fountains

O dataset “Drinkings Fountains” é baseado nas fontes existentes em Nova Iorque. Consiste em indicar as coordenadas geográficas, nome do bairro, tipo e ID de cada fonte existente na cidade de Nova Iorque.

Coluna 1: Objectid

Descrição: Identificação de cada fonte para o sistema.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.

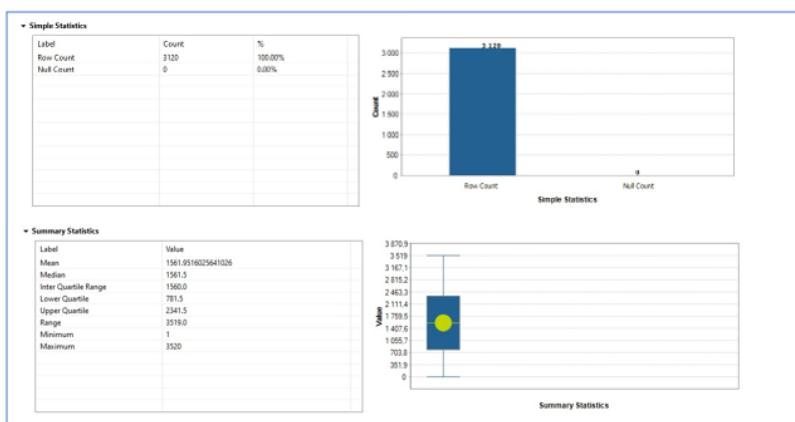


Figura 1 - Análise da coluna “Objectid”

Coluna 2: Fountain Type

Descrição: Tipo de fonte existente nos parques da cidade de nova iorque.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas de qualidade.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois há várias fontes no mesmo sítio.

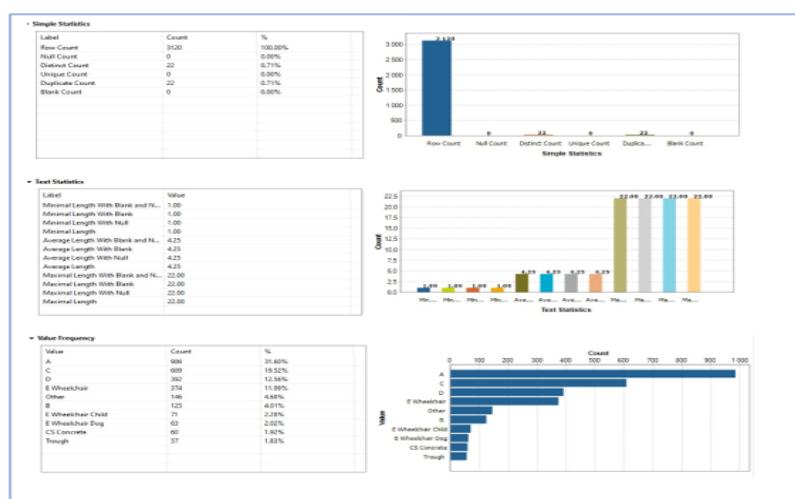


Figura 2 - Análise da coluna “Fountain Type”

Coluna 3: Position

Descrição: Descrição do ambiente do parque à volta da fonte.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco (Blank account)

Medidas Corretivas: Colocar como indefinido nas linhas em branco.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois há fontes em sítios iguais.



Figura 3 - Análise da coluna "Position"

Coluna 4: Collection Date

Descrição: Data do registo das fontes no sistema.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois as fontes foram adicionadas na mesma data.



Figura 4 - Análise da coluna "Collection Date"

Coluna 5: Painted

Descrição: Indica se a fonte está pintada.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Colocar como indefinido nas linhas em branco.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois fontes podem estar pintadas ou não.

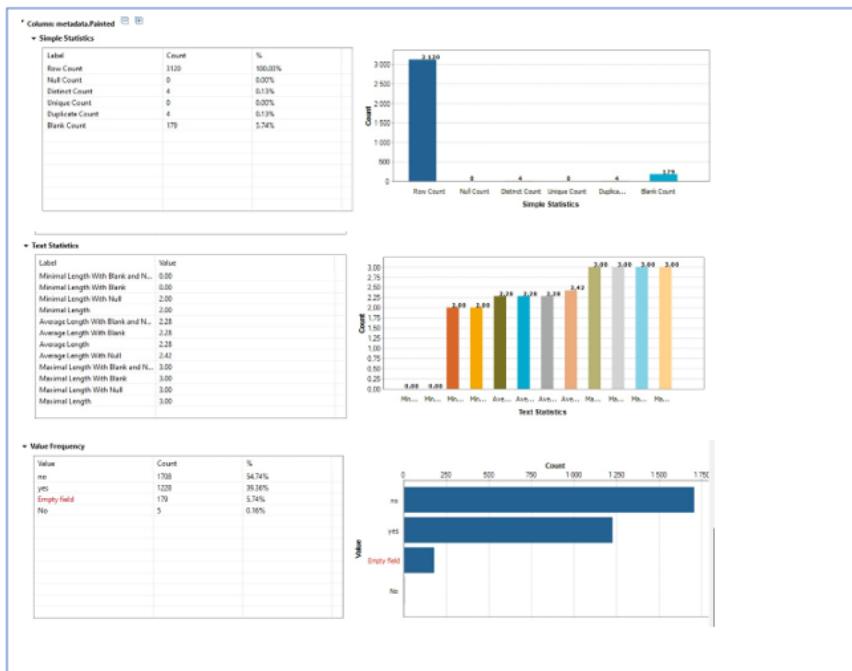


Figura 5 - Análise da coluna "Painted"

Coluna 6: Gispropnum

Descrição: Identificador único para a propriedade onde a fonte se localiza.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco, dados duplicados.

Medidas Corretivas: Colocar como indefinido nas linhas em branco, remover as linhas com dados duplicados.



Figura 6 - Análise da coluna "Gispropnum"

Coluna 7: Signname

Descrição: Nome da propriedade que a fonte se localiza.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois há várias fontes na mesma propriedade.



Figura 7 - Análise da coluna "Gispropnum"

Coluna 8: Borough

Descrição: Bairro que a fonte se localiza.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois há várias fontes no mesmo bairro.

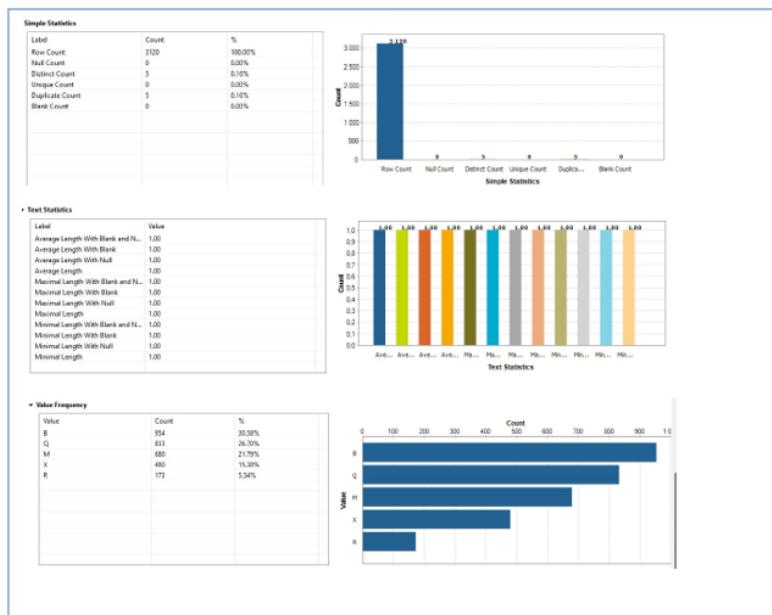


Figura 8 - Análise da coluna "Borough"

Coluna 9: Fountain Count

Descrição: Número de torneiras numa fonte.

Tipo de Dados: Int

Problemas de Qualidade: Não apresenta problemas.

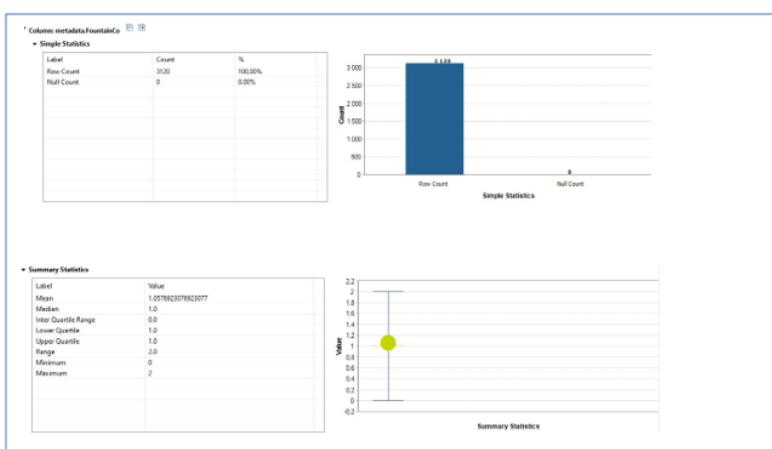


Figura 9 - Análise da coluna "Fountain Count"

Coluna 10: Gisobjid

Descrição: Identificador único para o sistema.

Tipo de Dados: Int

Problemas de Qualidade: Não apresenta problemas.



Figura 10 - Análise da coluna "Gisobjid"

Coluna 11: System

Descrição: Identificador único para o sistema.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.



Figura 11 - Análise da coluna "System"

Coluna 12: Department

Descrição: Permite identificar em que distrito do parque a fonte se localiza.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois há várias fontes no mesmo distrito do parque.



Figura 12 - Análise da coluna "Department"

Coluna 13: Parentid

Descrição: Identificador único para a propriedade onde a fonte se localiza.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco, dados duplicados.

Medidas Corretivas: Colocar como indefinido nas linhas em branco, remover as linhas com dados duplicados.



Figura 13 - Análise da coluna "Parentid"

Coluna 14: Description

Descrição: Descrição da fonte.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois há várias fontes no mesmo distrito do parque.



Figura 14 - Análise da coluna "Description"

Coluna 15: The geom

Descrição: Coordenadas das fontes nos parques de nova iorque.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas, dados duplicados

Medidas Corretivas: Remover as linhas com dados duplicados.



Figura 15 - Análise da coluna "Shape"

Coluna 16: Featurestatus

Descrição: Estado da fonte.

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas.

Nota: Embora apareçam dados duplicados, não há problemas de qualidade pois há várias fontes no mesmo distrito do parque.

Durante a análise do dataset, verificamos que dois atributos diferentes, contêm os mesmos dados (PARENTID e GISPROPNUM), assim iremos remover um desses atributos.

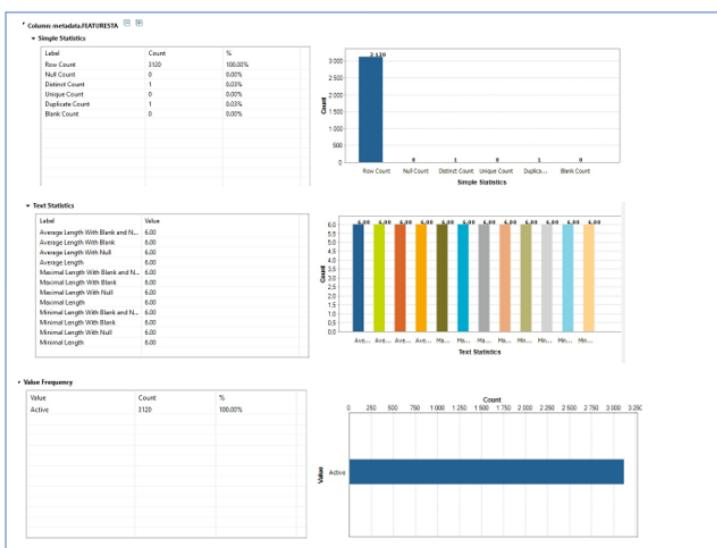


Figura 16 - Análise da coluna "Featurestatus"

Drinking Water Quality Distribution Monitoring

Este dataset resume os valores de turbidez, coliformes, flúor e cloro encontrados nas redes de distribuição em cada mês.

Coluna 1: Sample Number

Descrição: número de identificação da amostra.

Tipo de dados: Int

Problemas de Qualidade: Não apresenta problemas



Figura 17 - Análise da coluna "Sample Number"

Coluna 2: Sample Date

Descrição: data em que a amostra foi retirada

Tipo de Dados: String

Problemas de Qualidade: Não apresenta problemas de qualidade.

Nota: Esta coluna apresenta algumas linhas duplicadas, pois a data em que diferentes amostras foram retiradas pode coincidir.



Figura 18 - Análise da coluna "Sample Date"

Coluna 3: Sample Time

Descrição: hora em que a amostra foi retirada.

Tipo de dados: String

Problemas de Qualidade: não apresenta problemas

Nota: Esta coluna apresenta algumas linhas duplicadas, pois a hora em que diferentes amostras foram retiradas pode coincidir.



Figura 19 - Análise da coluna "Sample Time"

Coluna 4: Sample Site

Descrição: Local onde a amostra foi retirada.

Tipo de Dados: String

Problemas de Qualidade: não apresenta problemas

Nota: Esta coluna apresenta algumas linhas duplicadas, pois o local em que diferentes amostras foram retiradas pode coincidir.



Figura 20 - Análise da coluna "Sample Site"

Coluna 5: Sample Class

Descrição: Classe da amostra (Compliance, Operational, Resample compliance e Resample operational)

Tipo de dados: String

Problemas de Qualidade: Não apresenta.

Nota: Esta coluna apresenta algumas linhas duplicadas, pois a classe das diferentes amostras pode coincidir.

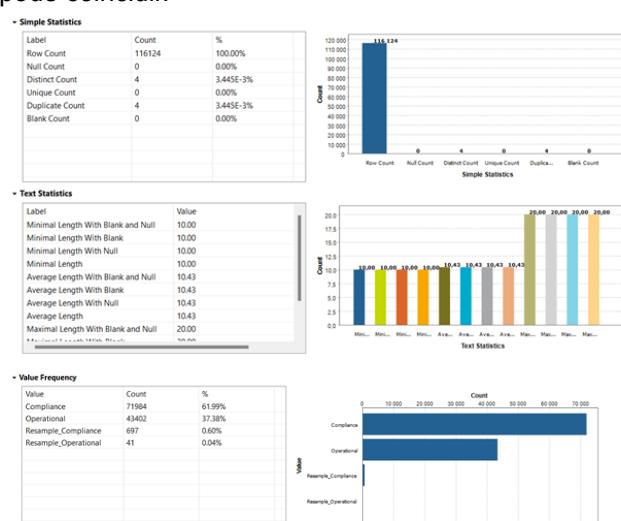


Figura 21 - Análise da coluna "Sample Class"

Coluna 6: Residual Free Chlorine

Descrição: Um nível de cloro livre de 0,5 mg/L é residual suficiente para manter a qualidade da água através da rede de distribuição, mas não é adequado para manter a qualidade da água quando esta é armazenada em casa por 24 horas.

Tipo de Dados: float

Problemas de Qualidade: Linhas com null.

Medidas Corretivas: Colocar “null”.

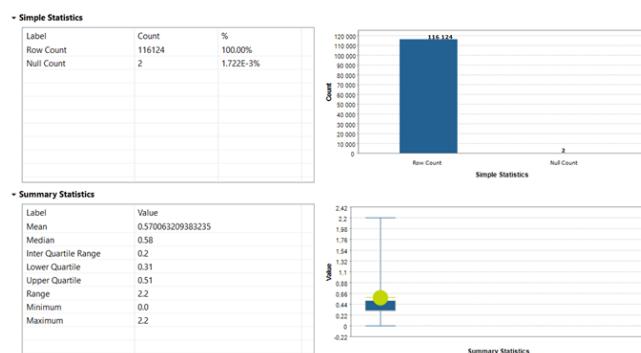


Figura 22 - Análise da coluna "Residual Free Chlorine"

Coluna 7: Turbidity

Descrição: propriedade física dos fluidos que se traduz na redução da sua transparência devido à presença de materiais em suspensão que interferem com a passagem da luz através do fluido.

Tipo de dados: float

Problemas de Qualidade: Linhas com null.

Medidas Corretivas: Colocar “null”.

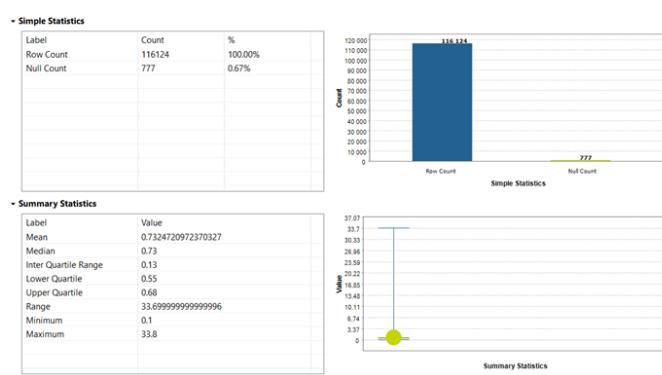


Figura 23 - Análise da coluna "Turbidity"

Coluna 8: Fluoride

Descrição: A fluorização da água é o ajustamento controlado do flúor a um abastecimento público de água apenas para reduzir a cárie dentária. O valor máximo de flúor permitido para que uma água seja considerada potável é 1,5 mgF/l

Tipo de dados: String

Problemas de Qualidade: Várias linhas em branco.

Medidas Corretivas: Como a percentagem de linhas em branco (86.94%) é consideravelmente grande, a melhor medida a tomar é eliminar esta coluna.



Figura 24 - Análise da coluna “Fluoride”

Coluna 9: Coliform Quantity Tray

Descrição: Coliformes são grupos de bactérias indicadoras de contaminação.

Tipo de dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Colocar “desconhecido” nas linhas em branco.

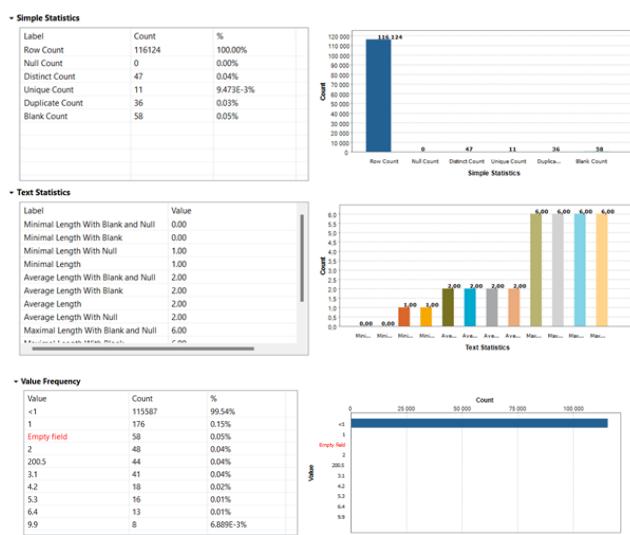


Figura 25 - Análise da coluna “Coliform Quantity Tray”

Coluna 10: E.Coli Quantity Tray

Descrição: Escherichia coli, é uma bactéria que se encontra normalmente no trato gastrointestinal inferior dos organismos de sangue quente. Para cada 100mL de água potável testada, não deve ser detetada E.coli;

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Colocar “desconhecido” nas linhas em branco.

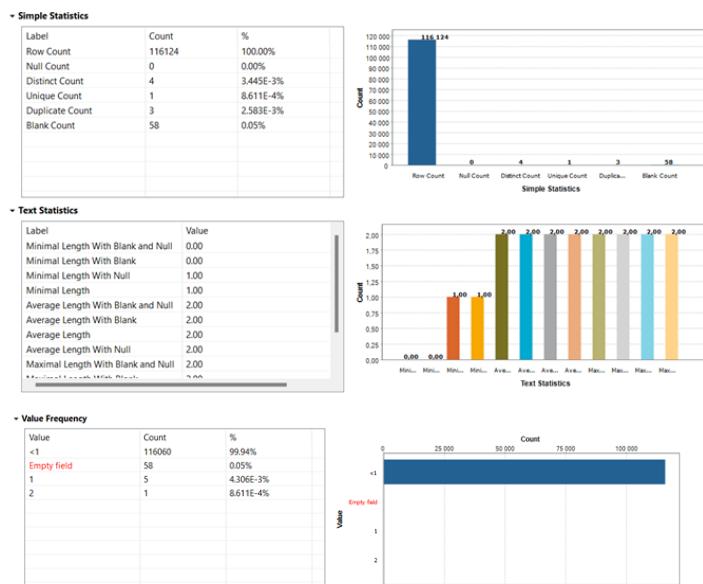


Figura 26 - Análise da coluna "E.Coli Quantity Tray"

DEP Green Infrastructure

O dataset “DEP Green Infrastructure” é baseado em locais da iniciativa de infraestruturas verdes de Nova Iorque, que apresenta uma abordagem alternativa para melhorar a qualidade da água que integra "infraestruturas verdes", tais como baleias e telhados verdes, com investimentos para otimizar o sistema existente e para construir infraestruturas "cinzentas" ou tradicionais específicas e rentáveis. As infraestruturas verdes recolhem águas pluviais das ruas, passeios e outras superfícies duras antes de poderem entrar no sistema de esgotos ou causar inundações locais. Ao reduzir a quantidade de águas pluviais que fluem para o sistema de esgotos, a infraestrutura verde ajuda a prevenir os transbordamentos de esgotos e melhora a saúde dos cursos de água locais.

Coluna 1: The Geom

Descrição: Coordenadas do ativo.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

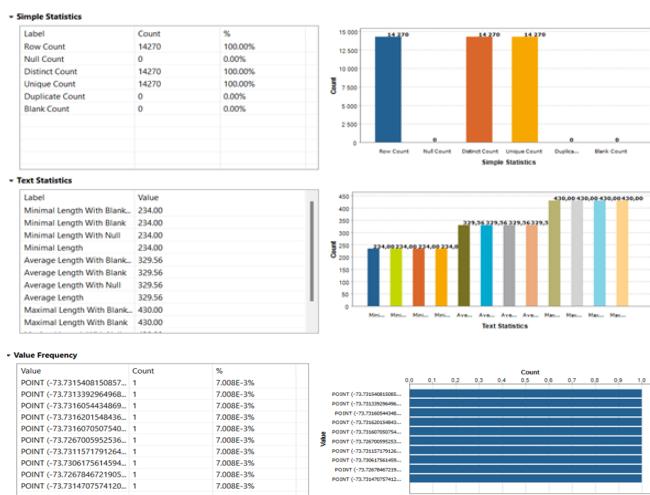


Figura 27 - Análise da coluna "The Geom"

Coluna 2: Asset_ID

Descrição: ID de ativo único

Tipo de Dados: Int

Problemas de Qualidade: Não tem problemas.



Figura 28 - Análise da coluna "Asset_ID"

Coluna 3: GI_ID

Descrição: ID único dentro de um contrato e fase.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

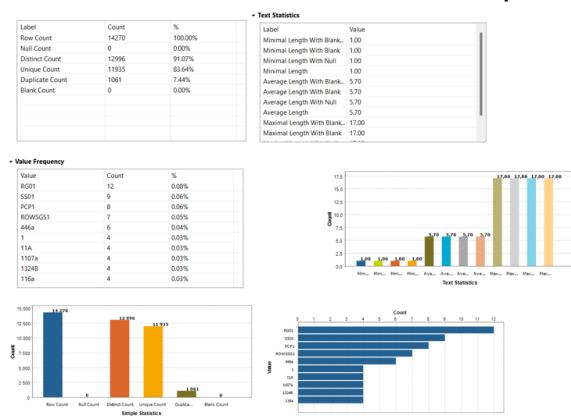


Figura 29 - Análise da coluna "GI_ID"

Coluna 4: DEP_Contra

Descrição: Número de contrato de design especificado pelo DEP para este projeto.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).

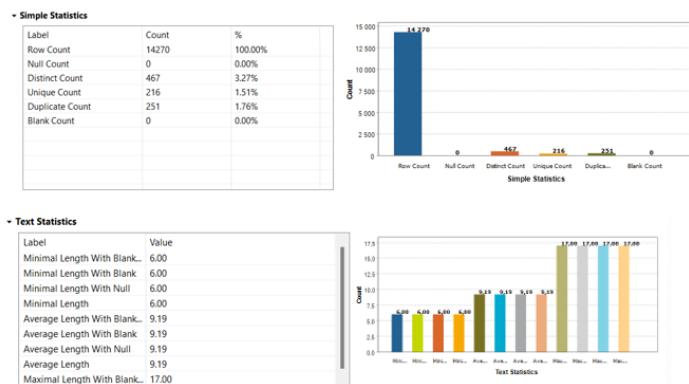


Figura 30 - Análise da coluna "DEP_Contra"

Coluna 5: Coluna DEP_Cont_1

Descrição: A designação de fase para o ativo, denotando o seu progresso dentro do contrato.

Tipo de Dados: Int

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Colocar null nas linhas em branco.

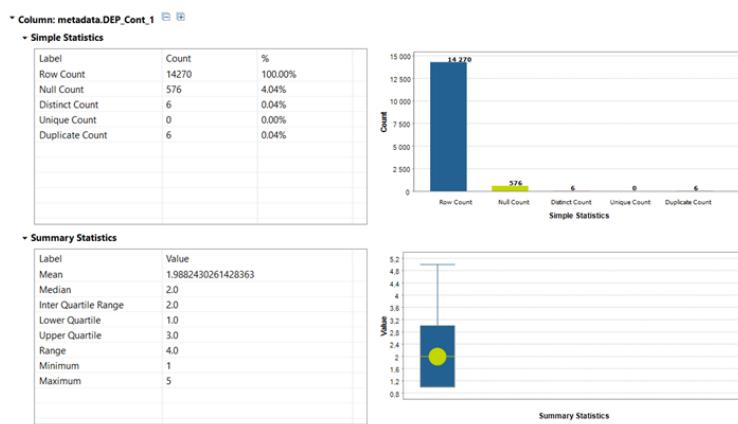


Figura 31 - Análise da coluna "DEP_Cont_1"

Coluna 6: Project_Na

Descrição: O nome do projeto que contém o contrato do ativo.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

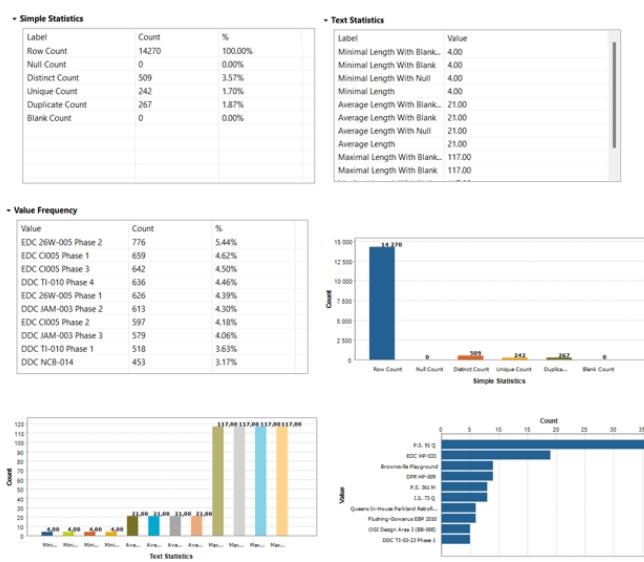


Figura 32 - Análise da coluna "Project_Na"

Coluna 7: ROW/Oncsite

Descrição: Projetos de direito de passagem (ROW) diferenciados de todos os outros projetos (Onsite).

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

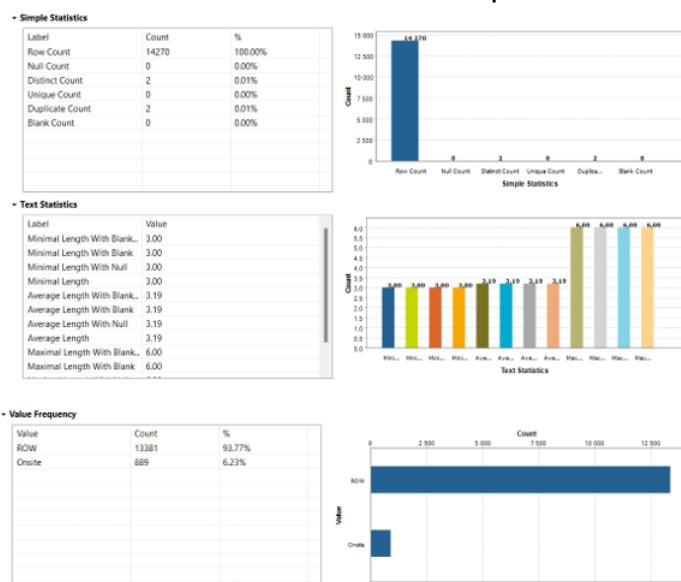


Figura 33 - Análise da coluna "ROW/Oncsite"

Coluna 8: Project_T

Descrição: A classe do projeto diz respeito à sua localização e à fonte de financiamento/conceção/propriedade da área afetada.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

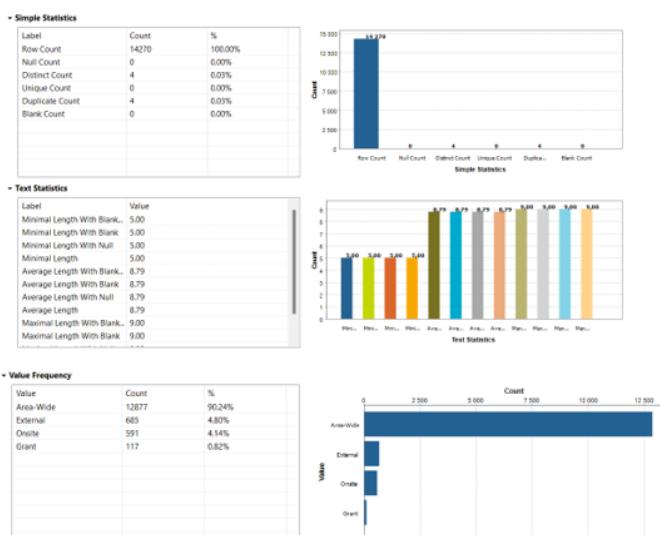


Figura 34 - Análise da coluna "Project_T"

Coluna 9: Asset_Type

Descrição: A classe de infraestrutura verde que define as especificidades do design e propósito de um ativo.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 35 - Análise da coluna "Asset_Type"

Coluna 10: Status

Descrição: Uma designação de nível que representa o estado de conceção do ativo e/ou conclusão da construção.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 36 - Análise da coluna "Status"

Coluna 11: Asset_X_Co

Descrição: A coordenada X ou longitudinal do ativo, no sistema de plano estatal EPSG #3104.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

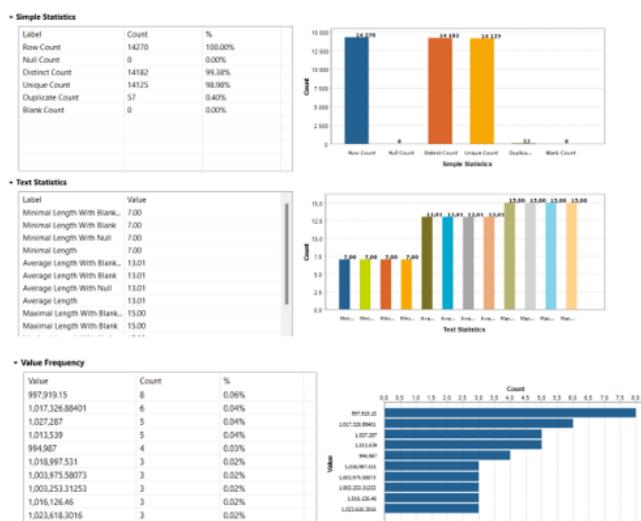


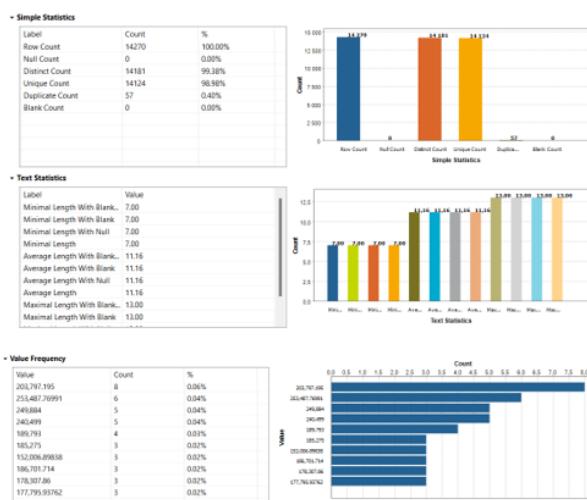
Figura 37 - Análise da coluna "Asset_X_Co"

Coluna 12: Asset_Y_Co

Descrição: A coordenada Y ou latitudinal do ativo, no sistema de plano estatal EPSG #3104.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Coluna 13: Borough

Descrição: Bairro de NYC em que o ativo está localizado.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Coluna 14: Sewer_Type

Descrição: O tipo de sistema de esgotos na área que contém o ativo.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

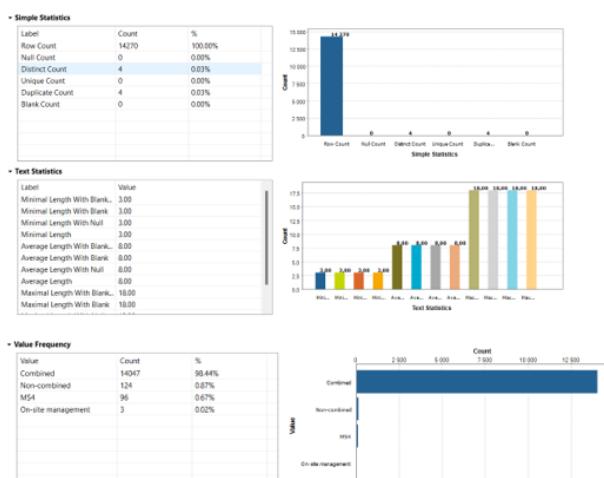


Figura 40 - Análise da coluna "Sewer_Type"

Coluna 15: Outfall

Descrição: O afluente em que o ativo está localizado.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 41 - Análise da coluna "Outfall"

Coluna 16: Waterbody

Descrição: O nome do corpo de água cuja bacia hidrográfica contém o ativo

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

**Figura 42 - Análise da coluna "Waterbody"**

Coluna 17: Nearest_In

Descrição: A rua mais próxima para o endereço da rua do ativo.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).

**Figura 43 - Análise da coluna "Nearest_In"**

Coluna 18: BBL

Descrição: Borough-block-lot valor que o ativo está dentro ou imediatamente frenteis.

Tipo de Dados: Int

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Colocar null nas linhas em branco.

**Figura 44 - Análise da coluna "BBL"**

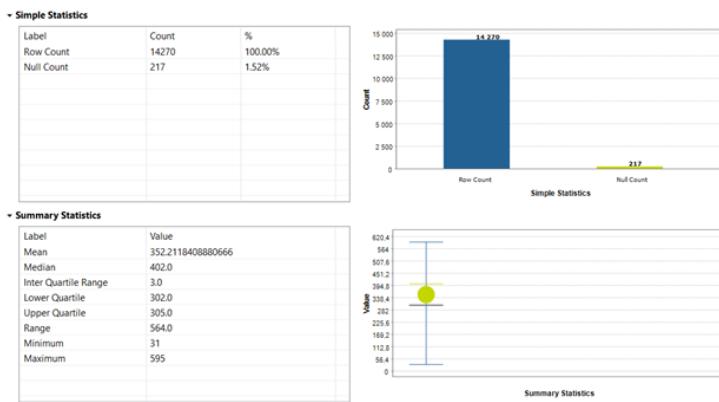
Coluna 19: Community

Descrição: O conselho comunitário no qual o ativo está localizado

Tipo de Dados: Int

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Colocar null nas linhas em branco.

**Figura 45 - Análise da coluna "Community"**

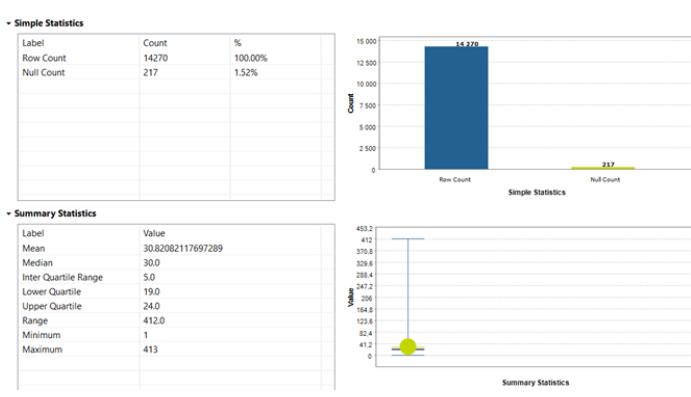
Coluna 20: City_Counc

Descrição: O distrito da câmara municipal em que o ativo está localizado

Tipo de Dados: Int

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Colocar null nas linhas em branco.

**Figura 46 - Análise da coluna "City_Counc"**

Coluna 21: Assembly_D

Descrição: O distrito de montagem em que o ativo está localizado.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).



Figura 47 - Análise da coluna "Assembly_D"

Coluna 22: Asset_Leng

Descrição: O comprimento de pegada do ativo nos pés.

Tipo de Dados: Float

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Colocar null nas linhas em branco.



Figura 48 - Análise da coluna "Asset_Leng"

Coluna 23: Asset_Width

Descrição: O comprimento de pegada do ativo nos pés.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

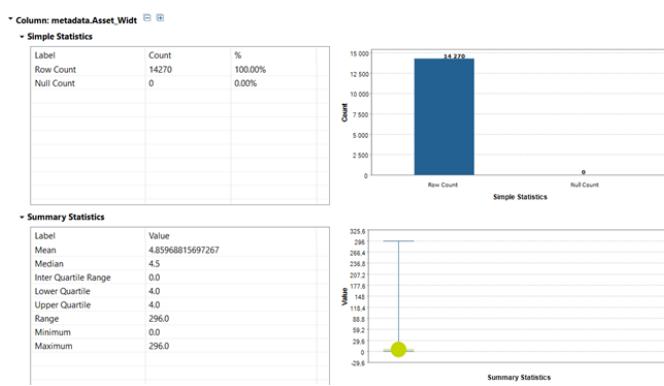


Figura 49 - Análise da coluna "Asset_Width"

Coluna 24: Asset_Area

Descrição: A área de pegada do ativo em metros quadrados.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 50 - Análise da coluna "Asset_Area"

Coluna 25: GI_Feature

Descrição: A categoria de infraestrutura verde a que o ativo pertence.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).

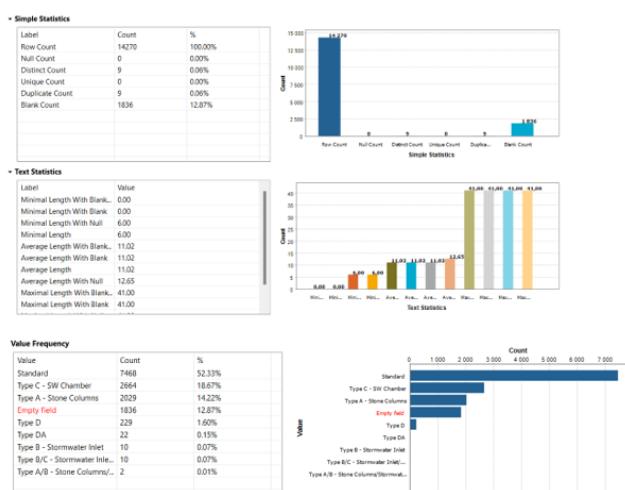


Figura 51 - Análise da coluna "GI_Feature"

Coluna 26: Tree_Latin

Descrição: Espécies com cultivar de árvore incluída no ativo, nome latino.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).



Figura 52 - Análise da coluna "Tree_Latin"

Coluna 27: Tree_Commo

Descrição: Espécies de árvores incluídas no ativo, nome comum.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).



Figura 53 - Análise da coluna "Tree_Como"

Coluna 28: Constructi

Descrição: Número do contrato de construção de DEP.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).



Figura 54 - Análise da coluna "Constructi"

Coluna 29: Construc_1

Descrição: Fase do contrato de construção do DEP.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).

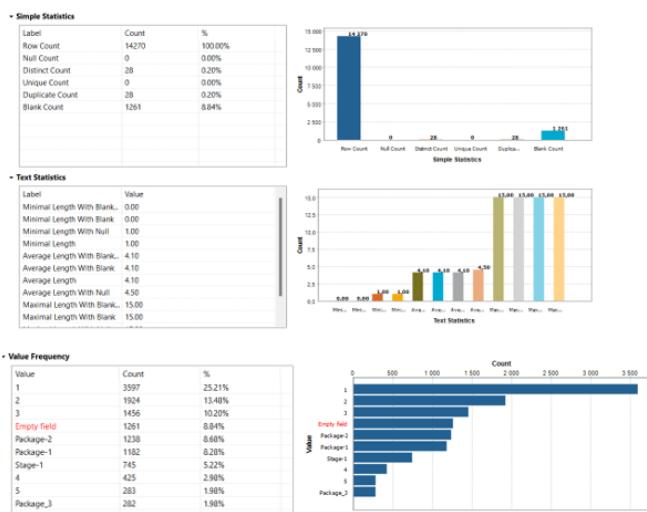


Figura 55 - Análise da coluna "Construc_1"

Coluna 30: Status_Gro

Descrição: Estado da construção do ativo.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 56 - Análise da coluna "Status_Gro"

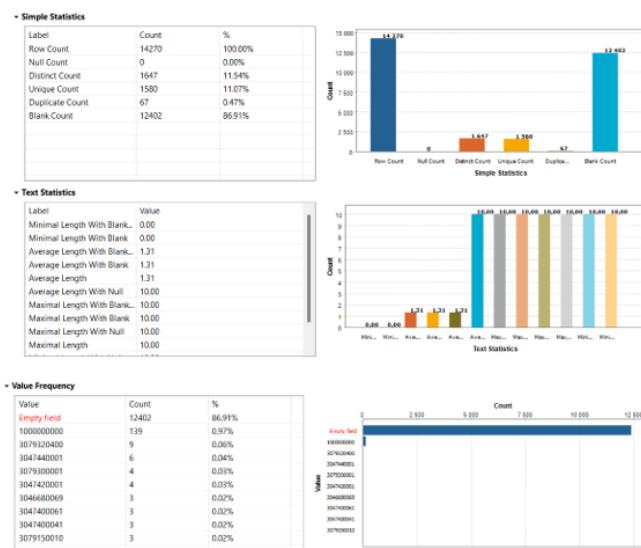
Coluna 31: Secondary_

Descrição: Segunda identificação do ativo.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco e irrelevância

Medidas Corretivas: Eliminação da coluna

*Figura 57 - Análise da coluna "Secondary_"*

Coluna 32: Street_Add

Descrição: O endereço físico mais próximo do bem.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco

Medidas Corretivas: Modificar os espaços em branco por Indefinido (Sem informação).

*Figura 58 - Análise da coluna "Street_Add"*

Water Tank Quality

Este dataset baseia-se na recolha de informação feita anualmente em tanques de água na cidade de Nova Iorque. Desde o ano 2000.

O dataset tem um total de 49 colunas das quais só utilizei 15 porque foram as colunas que achei que davam para relacionar com os datasets do resto do grupo e que continham mais informação sobre o tema que vamos abordar.

Coluna 1: Batch date

Descrição: Data do lote.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

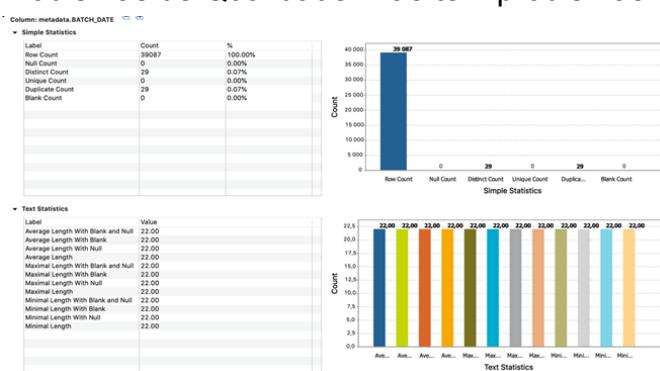


Figura 59 - Análise da coluna "Batch date"

Coluna 2: Bin

Descrição: Número de identificação do Department Buildings.

Tipo de Dados: String

Problemas de Qualidade: Linhas null.

Medidas Corretivas: Eliminar as linhas null.



Figura 60 - Análise da coluna "Bin"

Coluna 3: Block

Descrição: número do quarteirão onde está o tanque de água.

Tipo de Dados: String

Problemas de Qualidade: Linhas Duplicadas.

Medidas Corretivas: Eliminar as linhas duplicadas.

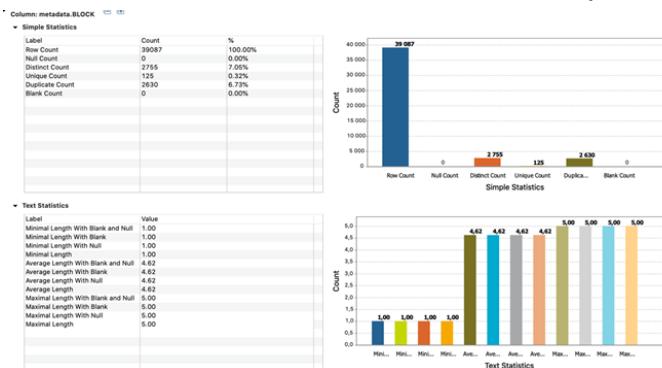


Figura 61 - Análise da coluna "Block"

Coluna 4: Borought

Descrição: Bairro onde se localiza o tanque

Tipo de Dados: String

Problemas de Qualidade: Linhas Duplicadas.

Medidas Corretivas: Eliminar as linhas duplicadas.

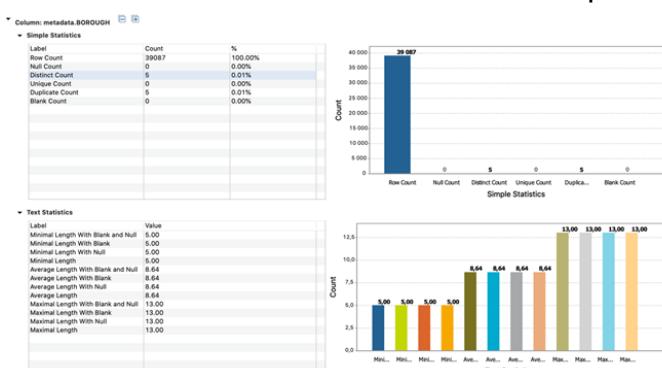


Figura 62 - Análise da coluna "Borought"

Coluna 5: Coliform

Descrição: Presença de coliform na água.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Eliminar as linhas em branco.

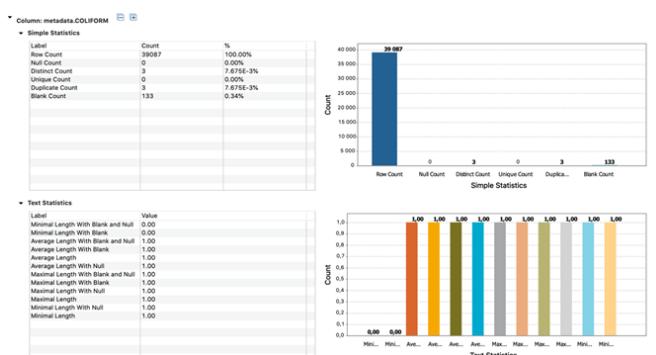


Figura 63 - Análise da coluna "Coliform"

Coluna 6: Ecoli

Descrição: Presença de ecoli na água.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Eliminar as linhas em branco.

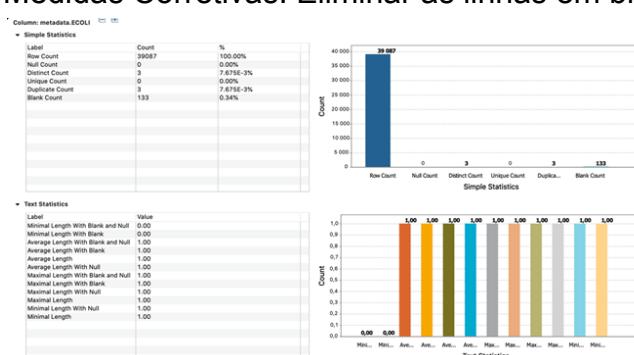


Figura 64 - Análise da coluna "Ecoli"

Coluna 7: House Number

Descrição: Número da rua onde se encontra o tanque.

Tipo de Dados: String

Problemas de Qualidade: Linhas duplicadas.

Medidas Corretivas: Eliminar as linhas duplicadas.

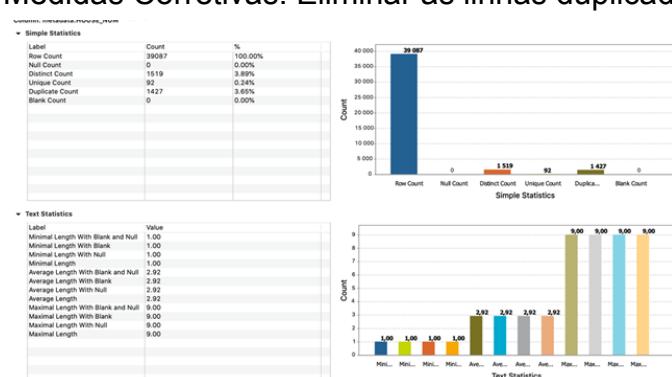


Figura 65 - Análise da coluna "House Number"

Coluna 8: Lab Name

Descrição: Nome do laboratório que fez a análise.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Eliminar as linhas em branco.

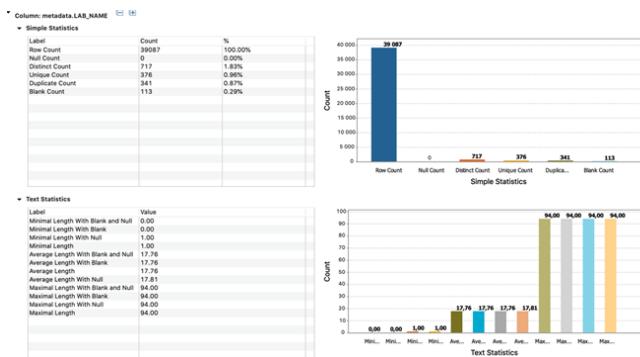


Figura 66 - Análise da coluna "Lab Name"

Coluna 9: Latitude

Descrição: latitude.

Tipo de Dados: Float

Problemas de Qualidade: Linhas com null.

Medidas Corretivas: Eliminar linhas com null.



Figura 67 - Análise da coluna "Latitude"

Coluna 10: Longitude

Descrição: Longitude.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.



Figura 68 - Análise da coluna "Longitude"

Coluna 11: Lot

Descrição: Número do lote.

Tipo de Dados: String

Problemas de Qualidade: Linhas duplicadas.

Medidas Corretivas: Eliminar as linhas duplicadas.

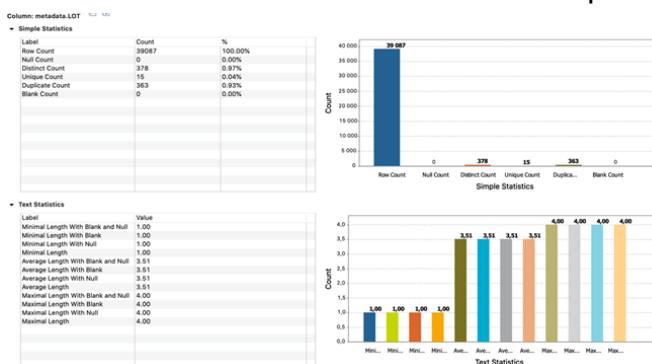


Figura 69 - Análise da coluna "Lot"

Coluna 12: Meet standards

Descrição: Padrões da qualidade da água.

Tipo de Dados: String

Problemas de Qualidade: Linhas em branco.

Medidas Corretivas: Eliminar as linhas em branco.

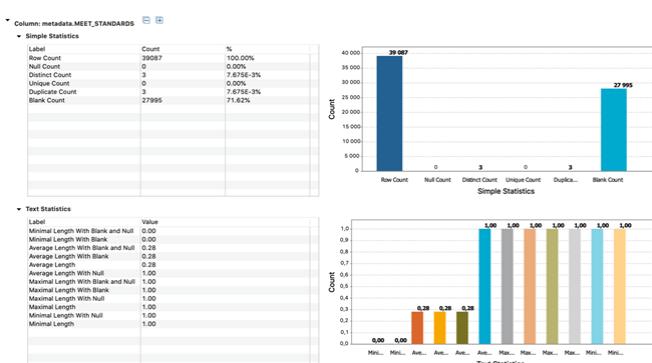


Figura 70 - Análise da coluna "Meet standards"

Coluna 13: Reporting year

Descrição: Ano em que foi feito o relatório.

Tipo de Dados: Int

Problemas de Qualidade: Não tem problemas.

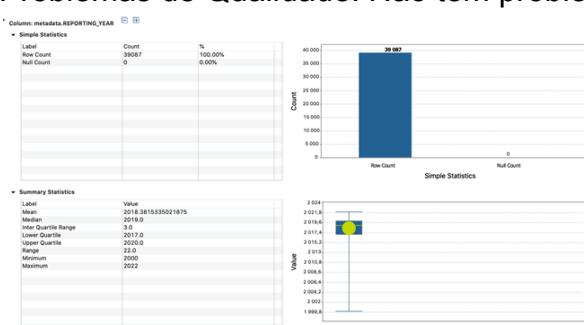


Figura 71 - Análise da coluna "Reporting year"

Coluna 14: Street name

Descrição: Nome da rua onde estão os tanques.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.

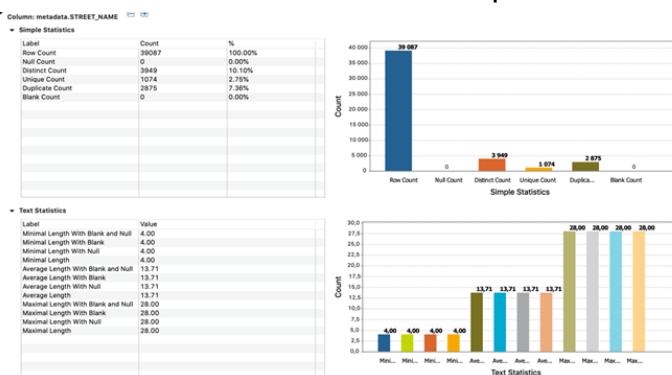


Figura 72 - Análise da coluna "Street name"

Coluna 15: Tank Number

Descrição: Número do tanque.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

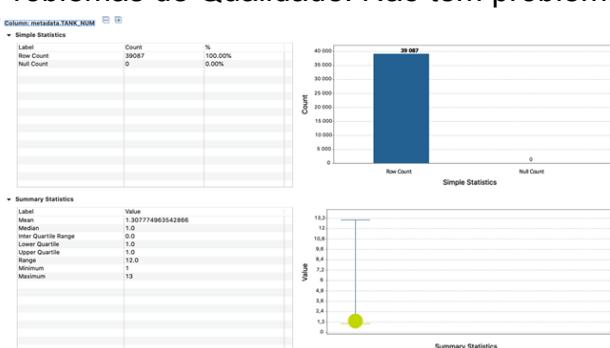


Figura 73 - Análise da coluna "Tank Number"

Harbor Water Quality

O dataset contém informação sobre a água do porto de NY.

Coluna 1: Sampling Location

Descrição: Localização da amostra.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 74 - Análise da coluna "Sampling Location"

Coluna 2: Corrent_Direction

Descrição: Direção da corrente.

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 75 - Análise da coluna "Corrent_Direction"

Coluna 3: Type

Descrição: Tipo(permanente)

Tipo de Dados: String

Problemas de Qualidade: Não tem problemas.



Figura 76 - Análise da coluna "Type"

Coluna 4: Top_Sample_Temperature

Descrição: Temperatura mais alta da amostra.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.



Figura 77 - Análise da coluna "Top_Sample_Temperature"

Coluna 5: Bottom_Sample_Temperature

Descrição: Temperatura mais baixa da amostra.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

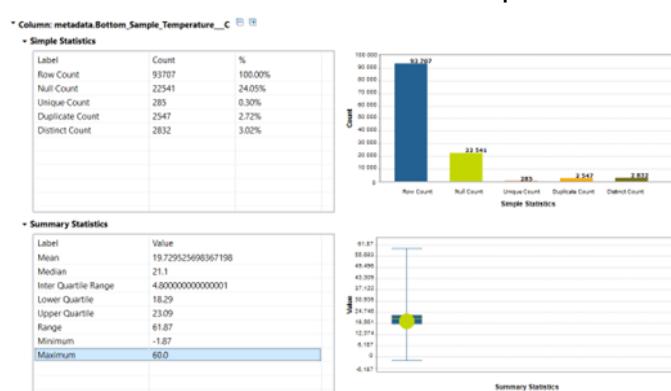


Figura 78 - Análise da coluna "Bottom_Sample_Temperature"

Coluna 6: Site_Actual_Depth_ft

Descrição: Profundidade da água do porto.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

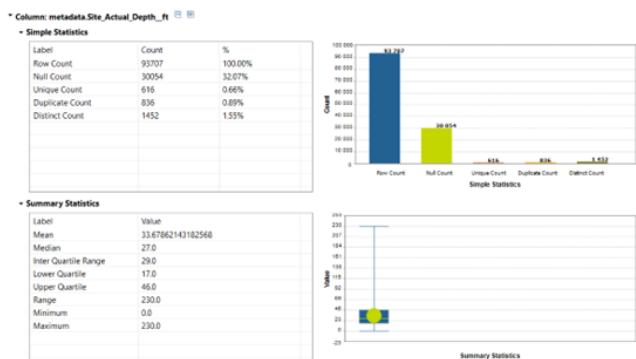


Figura 79 - Análise da coluna "Site_Actual_Depth_ft"

Coluna 7: Top_Salinity_psu

Descrição: Quantidade de sal existente na parte superior da água.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

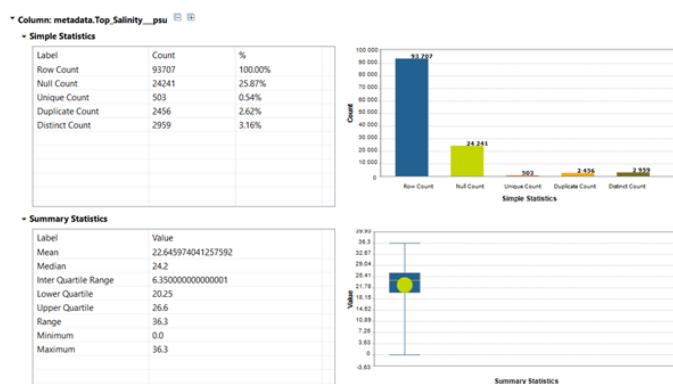


Figura 80 - Análise da coluna "Top_Salinity_psu"

Coluna 8: Bottom_Salinity_psu

Descrição: Quantidade de sal existente na parte mais funda da água.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.



Figura 81 - Análise da coluna "Bottom_Salinity_psu"

Coluna 9: Winkler_Top_Dissolved_O2

Descrição: oxigénio dissolvido à superfície da água do porto

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

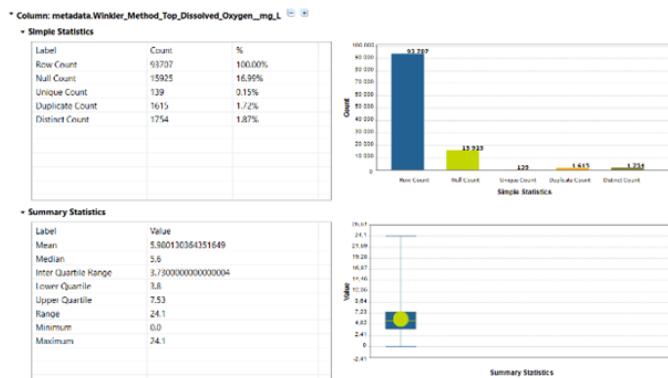


Figura 82 - Análise da coluna "Winkler_Top_Dissolved_O2"

Coluna 10: Winkler_Bottom_Dissolved_O2

Descrição: oxigénio dissolvido na parte mais funda do porto.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

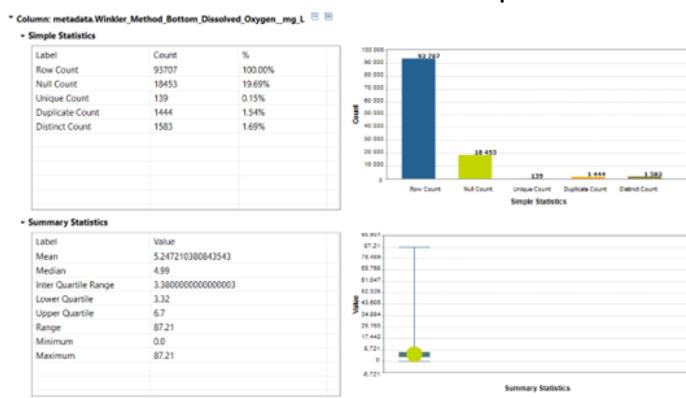


Figura 83 - Análise da coluna "Winkler_Bottom_Dissolved_O2"

Coluna 11: Long

Descrição: Longitude.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

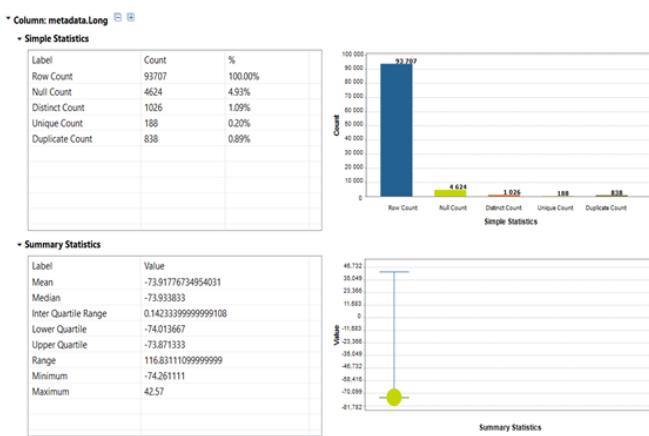


Figura 84 - Análise da coluna "Long"

Coluna 12: Lat

Descrição: Latitude.

Tipo de Dados: Float

Problemas de Qualidade: Não tem problemas.

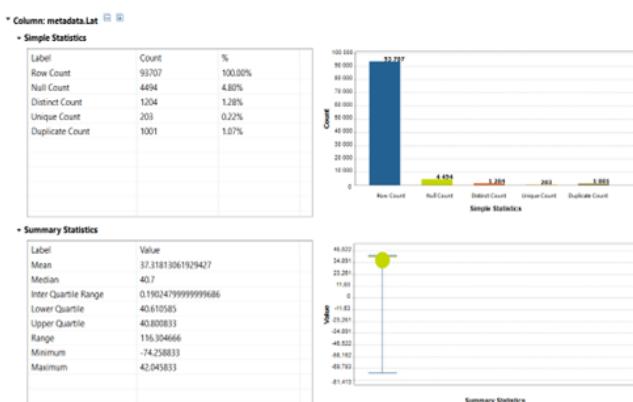


Figura 85 - Análise da coluna "Lat"

Coluna 13: Sample_data

Descrição: Data da recolha da amostra

Tipo de Dados: Date

Problemas de Qualidade: Há dados no dataset mas no talend não aparecem.

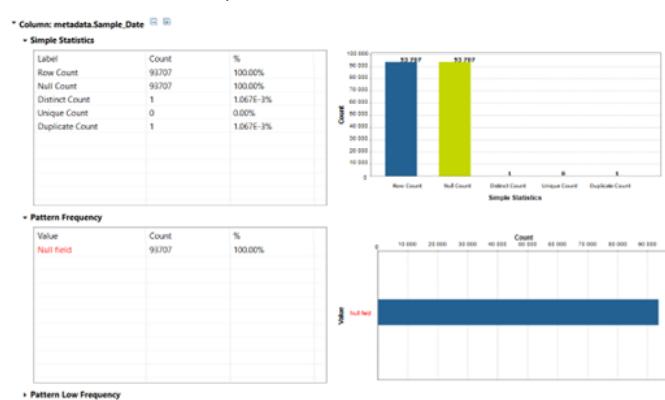


Figura 86 - Análise da coluna "Sample_data"

Coluna 14: Sample_Time

Descrição: Hora da recolha da amostra.

Tipo de Dados: Date

Problemas de Qualidade: Não tem problemas.



Figura 87 - Análise da coluna "Sample_Time"

Optamos pela remoção das restantes colunas devido ao facto das mesmas terem mais de 50% de linhas em branco e, sendo assim, consideramos que são dispensáveis.

Questões Analíticas

Drinking Fountains

1. Qual é o tipo de fonte mais comum nos parques de nova iorque (FountainType)?
2. A partir de 1899, todas as fontes existentes no bairro Q foram registadas no sistema (CollectionDate)?
3. Quantas fontes dentro do bairro Q (BOROUGH), são 'out in open'(DESCRIPTION)?
4. Qual é o bairro (BOROUGH) que apresenta mais fontes pintadas?

Drinking Water Quality Distribution Monitoring

1. Ao longo dos meses, a transparência da água (Turbidity) tem vindo a aumentar ou a diminuir?
2. Como é que o cloro (Residual Free Chlorine) influênciaria a qualidade da água através das redes de distribuição?
3. As bactérias (E.coli e Coliform) afetam os níveis de turbulência?
4. Os valores de cloro são influenciados pela hora em que as amostras são retiradas?

DEP Green Infrastructure

1. Como é que o desenvolvimento dos projetos DEP poderá ter melhorado a qualidade da água?
2. De que forma estes projetos afetam as espécies de árvores?
3. Qual foi a evolução da pegada ecológica ao longo dos projetos DEP existentes nas regiões?
4. Que fatores podem influenciar um projeto DEP a ser ativo?

Water Tank Quality

1. Desde o ano 2000 como foi a evolução da percentagem dos padrões da água dos tanques até 2022?
2. Com base na localização de Manhattan, os residentes são afetados por fraca qualidade de algum tanque de água?
3. A presença de coliform e/ou ecoli afeta os meet standards da água?
4. No ano 2016, quantos tanques foram reportados em Manhattan?
5. Quantos tanques com a presença de coliform existem em Manhattan?

Harbor Water Quality

1. De que forma a hora da recolha da amostra pode afetar a quantidade de sal existente na água?
2. De que maneira a quantidade de sal afeta a quantidade de oxigénio dissolvido na água?
3. A profundidade da água afeta a quantidade de oxigénio dissolvido?
4. De que maneira a hora da recolha da amostra afeta a temperatura da água?

KPI's

Drinking Fountains

1. Aumento das fontes registadas semestralmente (2%)
2. Aumento de fontes pintadas anualmente (2%)

Drinking Water Quality Distribution Monitoring

1. Diminuição, mensal, do nível de cloro na água (5%)
2. Diminuição, mensal, da turbulência da água (2%)
3. Aumento, mensal, da fluorização da água (3%)

DEP Green Infrastructure

1. Aumentar a percentagem de baleias e telhados verde (5%)
2. Diminuição do CSO (2%)
3. Aumento da pegada ecológica na área (3%)

Water Tank Quality

1. Baixar a temperatura da água em 3°C(2000-2018) em 18%
2. Aumentar a quantidade de oxigénio dissolvido na água em 10% (1980-2010)
3. Melhorar a proporção de tanques de água em Brooklyn(5.86%), bronx(4.66%), queens(3.92%) e staten Island(0.16%) comparativamente com Manhattan(85.39%).
4. Aumentar o número de relatórios em anos mais recentes.

Harbor Water Quality

1. Baixar a temperatura da água em 3°C(2000-2018) em 18%
2. Aumentar a quantidade de oxigénio dissolvido na água em 10% (1980-2010)

Arquitetura

Camada Bronze

Nesta primeira fase, faz-se a escolha do tema assim como a escolha dos datasets. Os dados estão muito brutos e a sua análise é de difícil compreensão. A partir desta etapa, a qualidade e o valor do dataset tendem a evoluir.

Para que isto seja possível, carregamos os datasets para o sistema de ficheiros HDFS. Neste sistema estruturaram-se os ficheiros da seguinte forma: “/demo/bronze/nome_dataset/ficheiro.csv”, como se pode ver nas figuras. O upload foi realizado através dos notebooks do Jupyter.

Drinking Fountains

```
from os import PathLike
from hdfs import InsecureClient
client = InsecureClient("http://hdfs-nn:9870/", user="anonymous")

from_path = "./DrinkingFountains.csv"
to_path = "/demo/bronze/Fontes/DrinkingFountains.csv"

client.delete(to_path)
client.upload(to_path, from_path)
```

Figura 88 - Código para colocar em bronze o dataset "DrinkingFountains"

Drinking Water Quality Distribution Monitoring

```
from os import PathLike
from hdfs import InsecureClient

client = InsecureClient("http://hdfs-nn:9870/", user="anonymous")

from_path = "./Drinking_Water_Quality_Distribution_Monitoring_Data.csv"
to_path = "/demo/bronze/Distribuicao_Agua/Drinking_Water_Quality_Distribution_Monitoring_Data.csv"

client.delete(to_path)
client.upload(to_path, from_path)
```

Figura 89 - Código para colocar em bronze o dataset "Drinking Water Quality Distribution Monitoring"

DEP Green Infrastructure

```
import sys
!{sys.executable} -m pip install hdfs

from os import PathLike
from hdfs import InsecureClient
client = InsecureClient("http://hdfs-nn:9870", user = "anonymous")

from_path = "./DEPGreenInfraestructure.csv"
to_path = "/demo/bronze/ProjetoGreen/DEPGreenInfraestructure.csv"

client.delete(to_path)
client.upload(to_path , from_path)
```

Figura 90 - Código para colocar em bronze o dataset "DEP Green Infraestructure"



Water Tank Quality

```
from os import PathLike
from hdfs import InsecureClient
client = InsecureClient("http://hdfs-nn:9870/", user="anonymous")

from_path = "./Self-Reported_Drinking_Water_Tank_Inspection_Results.csv"
to_path = "/demo/bronze/Agua_Tanques/Self-Reported_Drinking_Water_Tank_Inspection_Results.csv"

client.delete(to_path)

client.upload(to_path, from_path)
```

Figura 91 - Código para colocar em bronze o dataset "Water Tank Quality"

Harbor Water Quality

```
from os import PathLike
from hdfs import InsecureClient
client = InsecureClient("http://hdfs-nn:9870/", user="anonymous")

from_path = "./Harbor_Water_Quality.csv"
to_path = "/demo/bronze/Agua_Porto/Harbor_Water_Quality.csv"
client.delete(to_path)

client.upload(to_path, from_path)
```

Figura 92 - Código para colocar em bronze o dataset "Harbor Water Quality"

HDFS

Name	Size	Last Modified	Replication
Agua_Porto	0 B	Nov 10 15:06	0
Agua_Tanques	0 B	Nov 10 15:06	0
Distribucao_Agua	0 B	Nov 10 15:06	0
Fontes	0 B	Nov 10 15:06	0
ProjetoGreen	0 B	Nov 10 15:06	0

Figura 93 - Localização dos datasets no hdfs

Camada Silver

Nesta fase temos o primeiro passo de transformação dos dados. O dataset está mais acessível e a informação está mais clara e mais fácil de entender.

A camada silver consistia em carregar os dados que se encontravam na camada bronze para tabelas, realizando de seguida transformações mínimas, tais como a correção da qualidade de dados e a renomeação de colunas. Neste projeto utilizaram-se tabelas delta sugeridas pelos docentes.

Na análise da qualidade de dados, como se considerou que se iriam modificar as linhas que continham valores nulos e encontravam-se em branco para “NULL” e “Sem informação”, respectivamente.

Por fim, criou-se a base de dados, as tabelas para cada ficheiro, carregando-se posteriormente os dados de cada ficheiro para as tabelas.

Drinking Fountains

Na fase inicial, iniciamos o spark e em seguida apontamos para o local onde se encontra a nossa warehouse (Projeto/Silver).

```
from os import PathLike
from hdfs import InsecureClient
from pyspark.sql import SparkSession
from pyspark.sql import Row
from delta import *
from pyspark.sql.types import *
from pyspark.sql.functions import *

# warehouse_location points to the default location for managed databases and tables
warehouse_location = 'hdfs://hdfs-nn:9000/Projeto/Silver'

builder = SparkSession \
    .builder \
    .master("local[2]") \
    .appName("Python Spark DataFrames and SQL") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris", "thrift://hive-metastore:9083") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .config("spark.jars.packages", "io.delta:delta-core_2.12:1.0.0") \
    .enableHiveSupport() \
    .getOrCreate()

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

Nesta fase localizamos o nosso dataset e definimos o dataframe, colocando todas as colunas do dataset e o seu tipo, e assim, criamos uma tabela chamada “Fontes”.

```
hdfs_path = "hdfs://hdfs-nn:9000/Projeto/Bronze/DrinkingFountains.csv"

customSchema = StructType([
    StructField("FountainTy", StringType(), True),
    StructField("the_geom", StringType(), True),
    StructField("OBJECTID", StringType(), True),
    StructField("Position", StringType(), True),
    StructField("Collection", StringType(), True),
    StructField("Painted", StringType(), True),
    StructField("GISPROPNUM", StringType(), True),
    StructField("SIGNNAME", StringType(), True),
    StructField("BOROUGH", StringType(), True),
    StructField("FountainCo", IntegerType(), True),
    StructField("GISOBJID", IntegerType(), True),
    StructField("SYSTEM", StringType(), True),
    StructField("DEPARTMENT", StringType(), True),
    StructField("PARENTID", StringType(), True),
    StructField("DESCRIPTION", StringType(), True),
    StructField("FEATURESTA", StringType(), True),
])

])
Fontes = spark \
    .read \
    .option("delimiter", ",") \
    .option("header", "true") \
    .schema(customSchema) \
    .csv(hdfs_path)
Fontes.show()
Fontes.printSchema()
```

Nesta parte, eliminamos duas colunas do dataset ‘PARENTID’ e ‘FEATURESTA’.

```
replaced_Fontes = Fontes.drop("PARENTID")
replaced_Fontes2 = replaced_Fontes.drop("FEATURESTA")
replaced_Fontes2.toPandas()
```

Neste código estamos a acrescentar mais duas colunas, a coluna 'Data' que apresenta a data que a fonte foi colocada, e a coluna 'Year' que apresenta todos os anos que fontes foram registadas no sistema. Fizemos isto a partir de split da coluna “Collection”.

```
replaced_Fontes3 = replaced_Fontes2.withColumn('Data', split(replaced_Fontes2['Collection'], ' ').getItem(0))
replaced_Fontes3.toPandas()

replaced_Fontes4 = replaced_Fontes3.withColumn('Year', split(replaced_Fontes3['Data'], '/').getItem(2))
replaced_Fontes4.toPandas()
```

Agora substituímos por “Desconhecido” as linhas que se encontram vazias ou nulas nas colunas apresentadas: “Position”, “Painted” e “GISPROPNUM”.

```
replaced_Fontes5 = replaced_Fontes4.withColumn(
    "Position",
    when(
        (col("Position").isNull()),
        "Desconhecido"
    ).otherwise(col("Position")))

replaced_Fontes6 = replaced_Fontes5.withColumn(
    "Painted",
    when(
        (col("Painted").isNull()),
        "Desconhecido"
    ).otherwise(col("Painted")))

replaced_Fontes7 = replaced_Fontes6.withColumn(
    "GISPROPNUM",
    when(
        (col("GISPROPNUM").isNull()),
        "Indefinido"
    ).otherwise(col("GISPROPNUM")))
```

Neste dataset, no atributo ‘BOROUGH’ são apresentados os bairros de nova iorque mas apenas pela sua inicial, de maneira a facilitar a compreensão do dataset, este código serve para identificar o nome correto de cada bairro.



```
replaced_Fontes8 = replaced_Fontes7.withColumn(
    "BOROUGH",
    when(replaced_Fontes7.BOROUGH.endswith('B')).regexp_replace(replaced_Fontes7.BOROUGH,'B','Brooklyn')) \
.when(replaced_Fontes7.BOROUGH.endswith('M')).regexp_replace(replaced_Fontes7.BOROUGH,'M','Manhattan')) \
.when(replaced_Fontes7.BOROUGH.endswith('Q')).regexp_replace(replaced_Fontes7.BOROUGH,'Q','Queens')) \
.when(replaced_Fontes7.BOROUGH.endswith('R')).regexp_replace(replaced_Fontes7.BOROUGH,'R','Staten Island')) \
.when(replaced_Fontes7.BOROUGH.endswith('X')).regexp_replace(replaced_Fontes7.BOROUGH,'X','Bronx')) \
)
```

Nesta fase, criamos uma tabela externa na nossa database “Projeto” e dividimos por “Year” e por “BOROUGH”. A tabela está localizada em ‘`hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/DrinkingFountains`’.

```
spark.sql(
"""
CREATE EXTERNAL TABLE Projeto.DrinkingFountains (
    FountainTy VARCHAR (50),
    the_geom VARCHAR (70),
    OBJECTID VARCHAR (50),
    Position VARCHAR (50),
    Collection VARCHAR (50),
    Painted VARCHAR (5),
    GISPROPNUM VARCHAR (50),
    SIGNNAME VARCHAR (50),
    FountainCo INT,
    GISOBJID INT,
    SYSTEM VARCHAR (50),
    DEPARTMENT VARCHAR (50),
    DESCRIPTIO VARCHAR (50),
    Data DATE
)
USING DELTA
PARTITIONED BY (
    BOROUGH VARCHAR (50),
    Year INT
)
LOCATION 'hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/DrinkingFountains'
"""
)
)
```

Na ultima fase desta etapa, todas as tabelas e alterações feitas são guardadas na base de dados localizada em

‘`hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/DrinkingFountains/deltalake_table/`’

```
replaced_Fontes8 \
.select("FountainTy", "the_geom", "OBJECTID", "Position", "Collection", "Painted", "GISPROPNUM", "SIGNNAME", "FountainCo", "GISOBJID", "SYSTEM", "DI
.write \
.mode("overwrite") \
.partitionBy("BOROUGH", "Year") \
.format("delta") \
.save("hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/DrinkingFountains/deltalake_table/")
```

DEP Green Infrastructure

```
[31]: pip install delta-spark==2.1.1
```

```
Requirement already satisfied: delta-spark==2.1.1 in /opt/conda/lib/python3.10/site-packages (2.1.1)
Requirement already satisfied: importlib-metadata>=1.0.0 in /opt/conda/lib/python3.10/site-packages (from delta-spark==2.1.1) (4.11.4)
Requirement already satisfied: pyspark<3.4.0,>=3.3.0 in /usr/local/spark-3.3.0-bin-hadoop3/python (from delta-spark==2.1.1) (3.3.0)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.10/site-packages (from importlib-metadata>=1.0.0->delta-spark==2.1.1) (3.8.1)
Requirement already satisfied: py4j==0.10.9.5 in /opt/conda/lib/python3.10/site-packages (from pyspark<3.4.0,>=3.3.0->delta-spark==2.1.1) (0.10.9.5)
Note: you may need to restart the kernel to use updated packages.
```

```

from os import PathLike
from hdfs import InsecureClient
from pyspark.sql import SparkSession
from pyspark.sql import Row
from delta import *
from pyspark.sql.types import *
from pyspark.sql.functions import *

# warehouse_location points to the default location for managed databases and tables
warehouse_location = 'hdfs://hdfs-nn:9870/Projeto/silver/'

builder = SparkSession \
    .builder \
    .master("local[2]") \
    .appName("Python Spark DataFrames and SQL") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris", "thrift://hive-metastore:9083") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .config("spark.jars.packages", "io.delta:delta-core_2.12:2.1.1") \
    .enableHiveSupport() \

spark = configure_spark_with_delta_pip(builder).getOrCreate()

```

Inicializamos o spark e em seguida apontamos para o local onde se encontra a nossa warehouse , ou seja, no demo/silver.

```

: spark.sql(
    """
    DROP DATABASE Projeto CASCADE
    """
)

: DataFrame[]

: 
: spark.sql(
    """
    create database Projeto location 'hdfs://hdfs-nn:9000/Projeto/silver/Projeto.db'
    """
)

: DataFrame[]

: spark.sql(
    """
    SHOW TABLES FROM Projeto
    """
).show()

+-----+-----+-----+

```

Nesta parte criamos o nosso Projeto que será a nossa database (DataFrame) onde irão ser colocados as tabelas modificadas.

```

25]: hdfs_path = "hdfs://hdfs-nn:9000/demo/bronze/ProjetoGreen/DEPGreenInfraestructure.csv"

customSchema = StructType([
    StructField("the_geom", StringType(), True),
    StructField("Asset_ID", LongType(), True),
    StructField("GI_ID", StringType(), True),
    StructField("DEP_Contra", IntegerType(), True),
    StructField("DEP_Cont_1", IntegerType(), True),
    StructField("Project_Ty", StringType(), True),
    StructField("ROW_Onsite", StringType(), True),
    StructField("Project_Na", StringType(), True),
    StructField("Asset_Type", StringType(), True),
    StructField("Status", StringType(), True),
    StructField("Asset_X_Co", StringType(), True),
    StructField("Asset_Y_Co", StringType(), True),
    StructField("Borough", StringType(), True),
    StructField("Sewer_Type", StringType(), True),
    StructField("Outfall", StringType(), True),
    StructField("Waterbody", StringType(), True),
    StructField("Street_Add", StringType(), True),
    StructField("Nearest_In", StringType(), True),
    StructField("BBL", IntegerType(), True),
    StructField("Secondary_", StringType(), True),
    StructField("Community_", IntegerType(), True),
    StructField("City_Counc", IntegerType(), True),
    StructField("Assembly_D", StringType(), True),
    StructField("Asset_Leng", FloatType(), True),
    StructField("Asset_Width", FloatType(), True),
    StructField("Asset_Area", StringType(), True),
    StructField("GI_Feature", StringType(), True),
    StructField("Tree_Latin", StringType(), True),
    StructField("Tree_Como", StringType(), True),
    StructField("Constructi", StringType(), True),
    StructField("Construc_1", StringType(), True),
    StructField("Asset_Y_Co", StringType(), True),
    StructField("Borough", StringType(), True),
    StructField("Sewer_Type", StringType(), True),
    StructField("Outfall", StringType(), True),
    StructField("Waterbody", StringType(), True),
    StructField("Street_Add", StringType(), True),
    StructField("Nearest_In", StringType(), True),
    StructField("BBL", IntegerType(), True),
    StructField("Secondary_", StringType(), True),
    StructField("Community_", IntegerType(), True),
    StructField("City_Counc", IntegerType(), True),
    StructField("Assembly_D", StringType(), True),
    StructField("Asset_Leng", FloatType(), True),
    StructField("Asset_Width", FloatType(), True),
    StructField("Asset_Area", StringType(), True),
    StructField("GI_Feature", StringType(), True),
    StructField("Tree_Latin", StringType(), True),
    StructField("Tree_Como", StringType(), True),
    StructField("Constructi", StringType(), True),
    StructField("Construc_1", StringType(), True),
    StructField("Status_Gro", StringType(), True)
])

projeto_green = spark \
    .read \
    .option("delimiter", ",") \
    .option("header", "false") \
    .schema(customSchema) \
    .csv(hdfs_path)
projeto_green.show()
projeto_green.printSchema()

```

Nesta fase localizamos o nosso dataset e definimos o esquema para o dataframe, ou seja, colocamos todas as colunas do dataset e o seu tipo e criamos uma tabela chamada “projeto_green”.

```
[40]: replaced_projeto_green = projeto_green.drop("Secondary_")
replaced_projeto_green.toPandas()
```

Com este código eliminamos a coluna “Secondary_” e mostramos a tabela.

```
[42]: from pyspark.sql.functions import when, col, concat, lit
replaced2_projeto_green = replaced_projeto_green.withColumn(
    "DEP_Contra",
    when(
        (col("DEP_Contra").isNull() | (col("DEP_Contra") == None)),
        "Sem Informacao"
    ).otherwise(col("DEP_Contra")))
[43]: from pyspark.sql.functions import when, col, concat, lit
replaced3_projeto_green = replaced2_projeto_green.withColumn(
    "DEP_Cont_1",
    when(
        (col("DEP_Cont_1").isNull() | (col("DEP_Cont_1") == None)),
        "Null"
    ).otherwise(col("DEP_Cont_1")))
[44]: from pyspark.sql.functions import when, col, concat, lit
replaced4_projeto_green = replaced3_projeto_green.withColumn(
    "Nearest_In",
    when(
        (col("Nearest_In").isNull() | (col("Nearest_In") == None)),
        "Sem Informacao"
    ).otherwise(col("Nearest_In")))
[45]: from pyspark.sql.functions import when, col, concat, lit
replaced5_projeto_green = replaced4_projeto_green.withColumn(
    "BBL",
    when(
        (col("BBL").isNull() | (col("BBL") == None)),
        "Null"
```

```

replaced5_projeto_green = replaced4_projeto_green.withColumn(
    "BBL",
    when(
        (col("BBL").isNull() | (col("BBL") == None)),
        "Null"
    ).otherwise(col("BBL")))

from pyspark.sql.functions import when, col, concat, lit

replaced6_projeto_green = replaced5_projeto_green.withColumn(
    "Community_",
    when(
        (col("Community_").isNull() | (col("Community_") == None)),
        "Null"
    ).otherwise(col("Community_")))

from pyspark.sql.functions import when, col, concat, lit

replaced7_projeto_green = replaced6_projeto_green.withColumn(
    "City_Counc",
    when(
        (col("City_Counc").isNull() | (col("City_Counc") == None)),
        "Null"
    ).otherwise(col("City_Counc")))

from pyspark.sql.functions import when, col, concat, lit

replaced8_projeto_green = replaced7_projeto_green.withColumn(
    "Assembly_D",
    when(
        (col("Assembly_D").isNull() | (col("Assembly_D") == None)),
        "Sem Informacao"
    ).otherwise(col("Assembly_D")))

```

```

[49]: from pyspark.sql.functions import when, col, concat, lit

replaced9_projeto_green = replaced8_projeto_green.withColumn(
    "Asset_Leng",
    when(
        (col("Asset_Leng").isNull() | (col("Asset_Leng") == None)),
        "Null"
    ).otherwise(col("Asset_Leng")))

```

```

[50]: from pyspark.sql.functions import when, col, concat, lit

replaced10_projeto_green = replaced9_projeto_green.withColumn(
    "GI_Feature",
    when(
        (col("GI_Feature").isNull() | (col("GI_Feature") == None)),
        "Sem Informacao"
    ).otherwise(col("GI_Feature")))

```

```

[51]: from pyspark.sql.functions import when, col, concat, lit

replaced11_projeto_green = replaced10_projeto_green.withColumn(
    "Tree_Latin",
    when(
        (col("Tree_Latin").isNull() | (col("Tree_Latin") == None)),
        "Sem Informacao"
    ).otherwise(col("Tree_Latin")))

```

```
[52]: from pyspark.sql.functions import when, col, concat, lit
replaced12_projeto_green = replaced11_projeto_green.withColumn(
    "Tree_Como",
    when(
        (col("Tree_Como").isNull() | (col("Tree_Como") == "N/A")),
        "Sem Informacao"
    ).otherwise(col("Tree_Como")))
[53]: from pyspark.sql.functions import when, col, concat, lit
replaced13_projeto_green = replaced12_projeto_green.withColumn(
    "Constructi",
    when(
        (col("Constructi").isNull() | (col("Constructi") == None)),
        "Sem Informacao"
    ).otherwise(col("Constructi")))
[54]: from pyspark.sql.functions import when, col, concat, lit
replaced14_projeto_green = replaced13_projeto_green.withColumn(
    "Construc_1",
    when(
        (col("Construc_1").isNull() | (col("Construc_1") == None)),
        "Sem Informacao"
    ).otherwise(col("Construc_1")))
[54]: from pyspark.sql.functions import when, col, concat, lit
replaced14_projeto_green = replaced13_projeto_green.withColumn(
    "Construc_1",
    when(
        (col("Construc_1").isNull() | (col("Construc_1") == None)),
        "Sem Informacao"
    ).otherwise(col("Construc_1")))
[55]: from pyspark.sql.functions import when, col, concat, lit
replaced15_projeto_green = replaced14_projeto_green.withColumn(
    "Street_Add",
    when(
        (col("Street_Add").isNull() | (col("Street_Add") == None)),
        "Sem Informacao"
    ).otherwise(col("Street_Add")))
[56]: replaced15_projeto_green.toPandas()
```

Neste código, modificamos os espaços em branco por “Sem Informação” e “Null” no caso de ser inteiro.

```
[57]: spark.sql(
    """
    DROP TABLE IF EXISTS Projeto.DepGreenInfraestructure
    """
)
[57]: DataFrame[]
```

```
spark.sql(  
    """  
    CREATE EXTERNAL TABLE Projeto.DepGreenInfraestructure (  
        the_geom VARCHAR(50),  
        Asset_ID INT,  
        GI_ID VARCHAR(50),  
        DEP_Contra VARCHAR(50),  
        DEP_Cont_1 INT,  
        Project_Ty VARCHAR(50),  
        Row_Onsite VARCHAR(50),  
        Project_Na VARCHAR(50),  
        Asset_Type VARCHAR(50),  
        Status VARCHAR(50),  
        Asset_X_Co VARCHAR(50),  
        Asset_Y_Co VARCHAR(50),  
        Sewer_Type VARCHAR(50),  
        Outfall VARCHAR(50),  
        Waterbody VARCHAR(50),  
        Nearest_In VARCHAR(50),  
        BBL INT,  
        Community_ INT,  
        City_Counc INT,  
        Assembly_D VARCHAR(50),  
        Asset_Leng FLOAT,  
        Asset_Width FLOAT,  
        Asset_Area VARCHAR(50),  
        GI_Feature VARCHAR(50),  
        - - - - -  
        Status VARCHAR(50),  
        Asset_X_Co VARCHAR(50),  
        Asset_Y_Co VARCHAR(50),  
        Sewer_Type VARCHAR(50),  
        Outfall VARCHAR(50),  
        Waterbody VARCHAR(50),  
        Nearest_In VARCHAR(50),  
        BBL INT,  
        Community_ INT,  
        City_Counc INT,  
        Assembly_D VARCHAR(50),  
        Asset_Leng FLOAT,  
        Asset_Width FLOAT,  
        Asset_Area VARCHAR(50),  
        GI_Feature VARCHAR(50),  
        Tree_Latin VARCHAR(50),  
        Tree_Commo VARCHAR(50),  
        Constructi VARCHAR(50),  
        Construc_1 VARCHAR(50),  
        Status_Gro VARCHAR(50)  
    )  
    USING DELTA  
    PARTITIONED BY (  
        Borough VARCHAR(50),  
        Street_Add VARCHAR(50)  
    )  
    LOCATION 'hdfs://hdfs-nn:9000/Projeto/silver/Projeto.db/DepGreenInfraestructure'  
    """  
)
```

Criamos uma tabela externa chamada “DEPGreenInfraestructure” na nossa database “Projeto” e dividimos por “Borough” e “Street_Add” localizada no “`hdfs://hdfs-nn:9000/Projeto/silver/Projeto.db/DepGreenInfraestructure`”.

```
replaced15_projeto_green \
    .select("the_geom", "Asset_ID", "GI_ID", "DEP_Contra", "DEP_Cond_1", "Project_Ty", "Row_Onsite", "Project_Na", "Asset_Type", "Status", "Asset_X_Co", " \
    .write \
    .mode("overwrite") \
    .partitionBy("Borough", "Street_Add") \
    .format("delta") \
    .save("hdfs://hdfs-nn:9000/Projeto/silver/Projeto.db/deltalake_table")
```

Por fim, todas as tabelas e alterações feitas são guardadas na base de dados localizada em “`hdfs://hdfs-nn:9000/Projeto/silver/Projeto.db/deltalake_table`”.

Water Tank Quality

Iniciámos o spark e em seguida apontamos para o local onde se encontra a nossa warehouse (Projeto/Silver).

```
from os import PathLike
from hdfs import InsecureClient
from pyspark.sql import SparkSession
from pyspark.sql import Row
from delta import *
from pyspark.sql.types import LongType, StringType, StructField, StructType, BooleanType, ArrayType, IntegerType, FloatType
from pyspark.sql.functions import *

# warehouse_location points to the default location for managed databases and tables
warehouse_location = 'hdfs://hdfs-nn:9000/Projeto/Silver'

builder = SparkSession \
    .builder \
    .master("local[2]") \
    .appName("Python Spark DataFrames and SQL") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris", "thrift://hive-metastore:9083") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .config("spark.jars.packages", "io.delta:delta-core_2.12:1.0.0") \
    .enableHiveSupport()

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

Depois fazemos select das colunas que queremos.

```
hdfs_path = "hdfs://hdfs-nn:9000/Projeto/Bronze/Self-Reported_Drinking_Water_Tank_Inspection_Results.csv"

df_final = spark.read \
    .option("header", "true") \
    .option("delimiter", ",") \
    .csv(hdfs_path)

df = df_final.select ("BIN", "BOROUGH", "REPORTING_YEAR", "HOUSE_NUM", "STREET_NAME", "BLOCK", "LOT", "LAB_NAME", "COLIFORM", "ECOLI", "MEET_STANDARDS"
```

Nesta fase localizamos o nosso dataset e definimos o esquema para o dataframe, ou seja, colocamos todas as colunas do dataset e o seu tipo e criamos uma tabela chamada

“Agua_Tanques/Self_Reported_Drinking_Water_Tank_Inspection_Results”.

```
hdfs_path = "hdfs://hdfs-nn:9000/demo/bronze/Agua_Tanques/Self-Reported_Drinking_Water_Tank_Inspection_Results.csv"

customSchema = StructType([
    StructField("_BIN", IntegerType(), True),
    StructField("BOROUGH", StringType(), True),
    StructField("REPORTING_YEAR", IntegerType(), True),
    StructField("HOUSE_NUM", StringType(), True),
    StructField("STREET_NAME", StringType(), True),
    StructField("BLOCK", StringType(), True),
    StructField("LOT", StringType(), True),
    StructField("LAB_NAME", StringType(), True),
    StructField("COLIFORM", StringType(), True),
    StructField("ECOLI", StringType(), True),
    StructField("MEET_STANDARDS", StringType(), True),
    StructField("BATCH_DATE", StringType(), True),
    StructField("TANK_NUM", IntegerType(), True),
    StructField("LATITUDE", FloatType(), True),
    StructField("LONGITUDE", FloatType(), True),
])

AquaTanques = spark \
    .read \
    .option("delimiter", ",") \
    .option("header", "true") \
    .schema(customSchema) \
    .csv(hdfs_path)
AquaTanques.show()
AquaTanques.printSchema()
```

Depois começamos a tratar das colunas, trocando as linhas nulas para “Sem informação”

```
replaced_AquaTanques = AquaTanques.withColumn(
    "MEET_STANDARDS",
    when(
        (col("MEET_STANDARDS").isNull()),
        "Sem informação"
    ).otherwise(col("MEET_STANDARDS")))

replaced_AquaTanques2 = replaced_AquaTanques.withColumn(
    "LAB_NAME",
    when(
        (col("LAB_NAME").isNull()),
        "Sem informação"
    ).otherwise(col("LAB_NAME")))

replaced_AquaTanques3 = replaced_AquaTanques2.withColumn(
    "ECOLI",
    when(
        (col("ECOLI").isNull()),
        "Sem informação"
    ).otherwise(col("ECOLI")))

replaced_AquaTanques4 = replaced_AquaTanques3.withColumn(
    "COLIFORM",
    when(
        (col("COLIFORM").isNull()),
        "Sem informação"
    ).otherwise(col("COLIFORM")))
```

Criamos uma tabela externa chamada “Self_Reported_Drinking_Water_Tank_Inspection_Results” na nossa database “Projeto” e dividimos por “BOROUGH” e “REPORTING YEAR” localizada no “`hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Self_Reported_Drinking_Water_Tank_Inspection_Results`”.

```
spark.sql(
    """
    CREATE EXTERNAL TABLE Projeto.Self_Reported_Drinking_Water_Tank_Inspection_Results (
        _BIN INT,
        HOUSE_NUM VARCHAR(50),
        STREET_NAME VARCHAR(50),
        BLOCK VARCHAR(50),
        LOT VARCHAR(50),
        LAB_NAME VARCHAR(50),
        COLIFORM VARCHAR(1),
        ECOLI VARCHAR(1),
        MEET_STANDARDS VARCHAR(50),
        BATCH_DATE VARCHAR(50),
        TANK_NUM INT,
        LATITUDE FLOAT,
        LONGITUDE FLOAT
    )
    USING DELTA
    PARTITIONED BY (
        REPORTING_YEAR INT,
        BOROUGH VARCHAR(50)
    )
    )
    LOCATION 'hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Self_Reported_Drinking_Water_Tank_Inspection_Results'
"""
)
```

DataFrame[]

Por fim, todas as tabelas e alterações feitas são guardadas na base de dados localizada em

“`hdfs://hdfs-nn:9000/demo/silver/Projeto.db/Self_Reported_Drinking_Water_Inspection_Results/deltalake_table`”.

```
replaced_AguasTanques4 \
    .select("_BIN", "HOUSE_NUM", "STREET_NAME", "BLOCK", "LOT", "LAB_NAME", "COLIFORM", "ECOLI", "MEET_STANDARDS", "BATCH_DATE", "TANK_NUM", "LATITUDE", "LONGITUDE") \
    .write \
    .mode("overwrite") \
    .partitionBy("BOROUGH", "REPORTING_YEAR") \
    .format("delta") \
    .save("hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Self_Reported_Drinking_Water_Tank_Inspection_Results/deltalake_table/")
```

Harbor Water Quality

```
from os import PathLike
from hdfs import InsecureClient
from pyspark.sql import SparkSession
from pyspark.sql import Row
from delta import *
from pyspark.sql.types import *
from pyspark.sql.functions import *

# warehouse_location points to the default location for managed databases and tables
warehouse_location = 'hdfs://hdfs-nn:9000/Projeto/Silver'

builder = SparkSession \
    .builder \
    .master("local[2]") \
    .appName("Python Spark DataFrames and SQL") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris", "thrift://hive-metastore:9083") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .config("spark.jars.packages", "io.delta:delta-core_2.12:1.0.0") \
    .enableHiveSupport() \


spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

Aqui inicia-se a SparkSession

```
: spark.sql(  
    """  
    create database Projeto location 'hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db'  
    """  
)  
  
from pyspark.sql.types import StructType
```

Criar a base de dados no hdfs

```
hdfs_path = "hdfs://hdfs-nn:9000/Projeto/bronze/Harbor_Water_Quality.csv"  
  
Harbor_Water_Quality = spark \  
    .read \  
    .option("delimiter", ",") \  
    .option("header", "true") \  
    .option("inferSchema", True) \  
    .csv(hdfs_path)  
  
Harbor_Water_Quality.show()  
Harbor_Water_Quality.printSchema()  
Harbor_Water_Quality.toPandas()
```

Carregar a tabela para o dataframe

```
: replaced_hwq = Harbor_Water_Quality.drop("Duplicate Sample","Top Sample Depth(ft)","Bottom Sample Depth(ft)","Top Conductivity (S/m)","Bottom Conductiv  
replaced_hwq.printSchema()
```

Eliminar as colunas que não são necessárias

```

replaced_hwq1 = replaced_hwq.withColumn(
    "Sampling Location",
    when(
        (col("Sampling Location").isNull()),
        "Indefinido"
    ).otherwise(col("Sampling Location")))

replaced_hwq2 = replaced_hwq1.withColumn(
    "Sample Date",
    when(
        (col("Sample Date").isNull()),
        "Indefinido"
    ).otherwise(col("Sample Date")))

replaced_hwq3 = replaced_hwq2.withColumn(
    "Sample Time",
    when(
        (col("Sample Time").isNull()),
        "Indefinido"
    ).otherwise(col("Sample Time")))

replaced_hwq4 = replaced_hwq3.withColumn(
    "Top Sample Temperature (°C)",
    when(
        (col("Top Sample Temperature (°C)").isNull()),
        "Indefinido"
    ).otherwise(col("Top Sample Temperature (°C)")))

```

Substituir os valores nulos por Indefinido em todas as colunas

```

replaced_hwq15 = replaced_hwq14.withColumn('Year', split(replaced_hwq14['Sample Date'],'/').getItem(2))
replaced_hwq15.printSchema()
replaced_hwq15.show()

```

Criar a coluna Year para possibilitar as partições

```

NewColumns = (column.replace(' ','_') for column in replaced_hwq15.columns)
replaced_hwq16 = replaced_hwq15.toDF(*NewColumns)

replaced_hwq17 = replaced_hwq16.withColumnRenamed("Top_Sample_Temperature_(°C)","Top_Sample_Temperature_Celsius")
replaced_hwq18 = replaced_hwq17.withColumnRenamed("Bottom_Sample_Temperature_(°C)","Bottom_Sample_Temperature_Celsius")
replaced_hwq19 = replaced_hwq18.withColumnRenamed("Top_Salinity_(psu)","Top_Salinity_psu")
replaced_hwq20 = replaced_hwq19.withColumnRenamed("Bottom_Salinity_(psu)","Bottom_Salinity_psu")
replaced_hwq21 = replaced_hwq20.withColumnRenamed("Current_Direction_(Current_Direction)","Current_Direction")
replaced_hwq22 = replaced_hwq21.withColumnRenamed("Winkler_Method_Top_Dissolved_Oxygen_(mg/L)","Winkler_Method_Top_Dissolved_Oxygen_mgL")
replaced_hwq23 = replaced_hwq22.withColumnRenamed("Winkler_Method_Bottom_Dissolved_Oxygen_(mg/L)","Winkler_Method_Bottom_Dissolved_Oxygen_mgL")
replaced_hwq24 = replaced_hwq23.withColumnRenamed("Site_Actual_Depth_(ft)","Site_Actual_Depth_ft")

```

Retirar os espaços e caracteres especiais do nome das colunas

```

spark.sql(
    """
    CREATE EXTERNAL TABLE Projeto.Harbor_Water_Quality(
        Sampling_Location VARCHAR(100),
        Sample_Date VARCHAR(50),
        Sample_Time VARCHAR(50),
        Top_Sample_Temperature_Celsius VARCHAR(50),
        Bottom_Sample_Temperature_Celsius VARCHAR(50),
        Site_Actual_Depth_ft VARCHAR(50),
        Top_Salinity_psu VARCHAR(100),
        Bottom_Salinity_psu VARCHAR(100),
        Current_Direction VARCHAR(100),
        Winkler_Method_Top_Dissolved_Oxygen_mgL VARCHAR(100),
        Winkler_Method_Bottom_Dissolved_Oxygen_mgL VARCHAR(100),
        Long VARCHAR(50),
        Lat VARCHAR(100),
        Type VARCHAR(50)
    )
    USING DELTA
    PARTITIONED BY (
        Year VARCHAR(100)
    )
    LOCATION 'hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Harbor_Water_Quality'
"""
)
    
```

Criar a tabela no hdfs

```

replaced_hwq24.show()
replaced_hwq24 \
    .select("Sampling_Location", "Sample_Date", "Sample_Time", "Top_Sample_Temperature_Celsius", "Bottom_Sample_Temperature_Celsius", "Site_Actual_Depth_ft" \
            "Top_Salinity_psu", "Bottom_Salinity_psu", "Current_Direction", "Winkler_Method_Top_Dissolved_Oxygen_mgL", "Winkler_Method_Bottom_Dissolved_Oxy \
            "Lat", "Type", "Year") \
    .write \
    .mode("overwrite") \
    .partitionBy("Year") \
    .format("delta") \
    .save("hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Harbor_Water_Quality")
    
```

Carregar os dados do dataframe para a tabela

Drinking Water Quality Distribution Monitoring

Iniciámos o spark e em seguida apontamos para o local onde se encontra a nossa warehouse (Projeto/Silver).

```
from os import PathLike
from hdfs import InsecureClient
from pyspark.sql import SparkSession
from pyspark.sql import Row
from delta import *
from pyspark.sql.types import LongType, StringType, StructField, StructType, BooleanType, ArrayType, IntegerType, FloatType
from pyspark.sql.functions import *

# warehouse_location points to the default location for managed databases and tables
warehouse_location = 'hdfs://hdfs-nn:9000/Projeto/Silver'

builder = SparkSession \
    .builder \
    .master("local[2]") \
    .appName("Python Spark DataFrames and SQL") \
    .config("spark.sql.warehouse.dir", warehouse_location) \
    .config("hive.metastore.uris", "thrift://hive-metastore:9083") \
    .config("spark.sql.extensions", "io.delta.sql.DeltaSparkSessionExtension") \
    .config("spark.sql.catalog.spark_catalog", "org.apache.spark.sql.delta.catalog.DeltaCatalog") \
    .config("spark.jars.packages", "io.delta:delta-core_2.12:1.0.0") \
    .enableHiveSupport() \

spark = configure_spark_with_delta_pip(builder).getOrCreate()
```

```
spark.sql(
"""
create database Projeto location 'hdfs://hdfs-nn:9000/demo/silver/Projeto.db'
"""
)
```

```
hdfs_path = "hdfs://hdfs-nn:9000/Projeto/Bronze/Drinking_Water_Quality_Distribution_Monitoring_Data.csv"

customSchema = StructType([
    StructField("Sample_Number", IntegerType(), True),
    StructField("Sample_Date", StringType(), True),
    StructField("Sample_Time", StringType(), True),
    StructField("Sample_Site", StringType(), True),
    StructField("Sample_class", StringType(), True),
    StructField("Residual_Free_Chlorine__mg_L", FloatType(), True),
    StructField("Turbidity__NTU", FloatType(), True),
    StructField("Fluoride__mg_L", StringType(), True),
    StructField("Coliform__Quanti_Tray__MPN_100mL", StringType(), True),
    StructField("Ecoli__Quanti_Tray__MPN_100mL", StringType(), True),
])

drinkingWQuality = spark \
    .read \
    .option("delimiter", ",") \
    .option("header", "true") \
    .schema(customSchema) \
    .csv(hdfs_path)
drinkingWQuality.show()
drinkingWQuality.printSchema()
```

Aqui, eliminámos a coluna “Fluorido__mg_L”.

```
replaced_drinkingWQuality = drinkingWQuality.drop("Fluoride__mg_L")
replaced_drinkingWQuality.toPandas()
```

Neste código estamos a acrescentar mais uma coluna que vai conter o ano em que a amostra foi retirada. Fazendo split da coluna “Sample_Date”.

```
replaced_drinkingWQuality2 = replaced_drinkingWQuality.withColumn('Year', split(replaced_drinkingWQuality['Sample_Date'], '/').getItem(2))
replaced_drinkingWQuality2.toPandas()
```

Agora substituímos por “null” (número) ou “Sem Informação” (string) as linhas que se encontram vazias ou nulas nas colunas “Residual_Free_Chlorine__mg_L”, “Turbidity__NTU”, “Coliform__Quanti_Tray_MPN_100ml” e “Ecoli_Quanti_Tray_MPN_100ml”

```
replaced_drinkingWQuality3 = replaced_drinkingWQuality2.withColumn(
    "Residual_Free_Chlorine__mg_L",
    when(
        (col("Residual_Free_Chlorine__mg_L").isNull()),
        "null"
    ).otherwise(col("Residual_Free_Chlorine__mg_L")))

replaced_drinkingWQuality4 = replaced_drinkingWQuality3.withColumn(
    "Turbidity__NTU",
    when(
        (col("Turbidity__NTU").isNull()),
        "null"
    ).otherwise(col("Turbidity__NTU")))

replaced_drinkingWQuality5 = replaced_drinkingWQuality4.withColumn(
    "Coliform__Quanti_Tray__MPN_100ml",
    when(
        (col("Coliform__Quanti_Tray__MPN_100ml").isNull()),
        "Sem informação"
    ).otherwise(col("Coliform__Quanti_Tray__MPN_100ml")))

replaced_drinkingWQuality6 = replaced_drinkingWQuality5.withColumn(
    "Ecoli_Quanti_Tray__MPN_100ml",
    when(
        (col("Ecoli_Quanti_Tray__MPN_100ml").isNull()),
        "Sem informação"
    ).otherwise(col("Ecoli_Quanti_Tray__MPN_100ml")))
```

Criamos uma tabela externa na nossa database “Projeto” e dividimos por “Year”. Localizada em

'hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Drinking_Water_Quality_Distribution_Monitoring_Data'.

```
spark.sql(
"""
CREATE EXTERNAL TABLE Projeto.Drinking_Water_Quality_Distribution_Monitoring_Data (
    Sample_Number INT,
    Sample_Date VARCHAR(50),
    Sample_Time VARCHAR(50),
    Sample_Site VARCHAR(50),
    Sample_class VARCHAR(50),
    Residual_Free_Chlorine__mg_L INT,
    Turbidity__NTU INT,
    Coliform__Quanti_Tray__MPN_100ml VARCHAR(50),
    Ecoli_Quanti_Tray__MPN_100ml VARCHAR(50)
)
USING DELTA
PARTITIONED BY (
    Year INT
)
LOCATION 'hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Drinking_Water_Quality_Distribution_Monitoring_Data'
"""
)
```

Finalmente, todas as tabelas e alterações feitas guardadas na base de dados localizada em

"hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Drinking_Water_Quality_Distribution_Monitoring_Data/deltalake_table/"

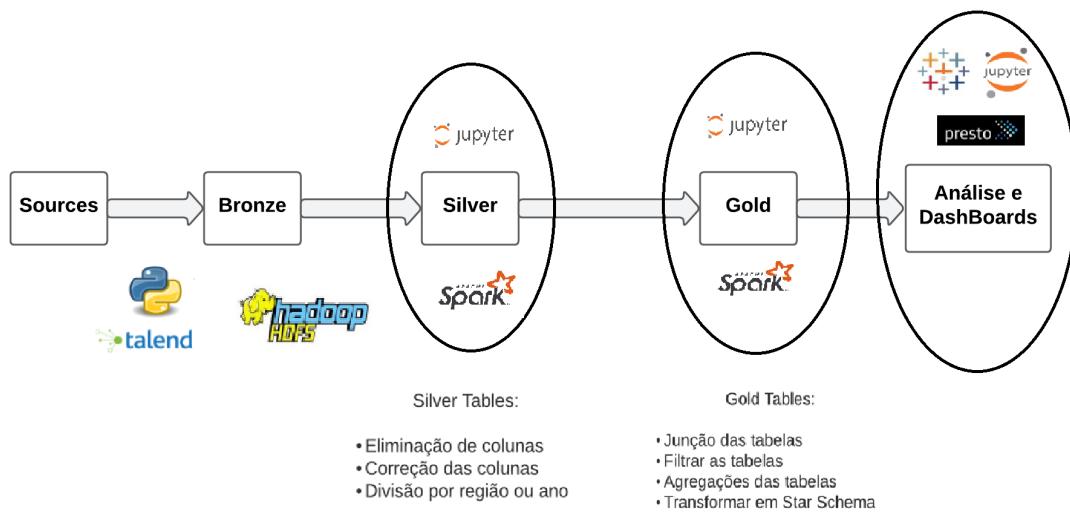
```
replaced_drinkingWQuality6 \
.select("Sample_Number", "Sample_Date", "Sample_Time", "Sample_Site", "Sample_class", "Residual_Free_Chlorine__mg_L", "Turbidity__NTU", "Coliform__Quanti_Tray__MPN_100ml") \
.write \
.mode("overwrite") \
.partitionBy("Year") \
.format("delta") \
.save("hdfs://hdfs-nn:9000/Projeto/Silver/Projeto.db/Drinking_Water_Quality_Distribution_Monitoring_Data/deltalake_table/")
```

Camada Gold

É a camada final e nesta etapa os dados já passaram por uma transformação total. A informação já está pronta para ser apresentada e o seu valor está num nível muito mais elevado do que estava na fase bronze.

Nesta fase iremos fazer a junção das tabelas de forma a responder às questões analíticas e criar uma tabela gold e presto.

Diagrama Pipeline



Conclusão

Os objetivos para este projeto foram definir o tema, determinar os respectivos datasets, questões analíticas e KPIs, análise dos dados, extração, carregamento e transformação dos dados.

A cada semana realizou-se pelo menos uma reunião para ponto de situação, esclarecimento de dúvidas e determinação das próximas tarefas a realizar, de modo a conseguir um projeto coerente.

Para a análise dos dados utilizou-se o Talent Open Studio que permitiu saber quais os tipos de erros que existiam nos dados obtidos e informações existentes em cada dataset. Para a extração, carregamento e transformação dos dados utilizaram-se as aplicações Docker, Jupyter, PySpark, HDFS e Hive.