Database Basics

What is a RDBMS?

- Edgar F. Codd (IBM, 1960's) invented the idea of the first RDBMS which improved on the hierarchical model (think bio. taxonomy) for efficiently storing data.
- A relational database management system (RDBMS) is a program used to manage a relational database.
- A relational database stores "relations" consisting of a set of tuples with shared attributes. You're familiar with these as tables with rows and columns.

  Record 1 (Attribute 1, Attribute 2, … Attribute N)

  Record 2 (Attribute 1, Attribute 2, … Attribute N)

  …

  Record R (Attribute 1, Attribute 2, … Attribute N)
- RDBMS is the basis for modern Structured Query Language (SQL) databases.

Relationships

- **One-to-one**: Each record in Table A relates to *only one* record in Table B and vice versa.
- **One-to-many**: A record in Table A can relate to zero, one, or many records in Table B. Each record in Table B relates to *only one* record in Table A.
- **Many-to-many**: A record in Table A can relate to zero, one, or many records in Table B. A record in Table B can relate to zero, one, or many records in Table A.

Tables

- Tables represent fundamental associations of data for the problem space.
- A column is the smallest organization structure within the table.
- A row represents a unique instance of the relationship a table holds.
- A **Primary Key** differentiates unique instances within a table.

Bidder

Bidder_ID	Bidder Name
235	Dutra Dredging Co.
622	Weeks Marine, Inc (ATLANTIC)
24	Southern Dredging Co., Inc.

Primary Key

Tables

- A **Foreign Key** relates entities within one relationship (the parent) to another relationship (the child).

Parent

Bidder

Primary Key

Bidder_ID		Bidder Name
235		Dutra Dredging Co.
622		Weeks Marine, Inc (ATLANTIC)
24		Southern Dredging Co., Inc.

Child

Plant

Foreign Key

Plant_ID	Plant Name	Plant Type	Bidder_ID
1	Stuyvesant	Hopper	235
2	Weeks	Hopper	622
3	Brunswick	Pipeline	24

Tables

A **Global ID** is a unique identifier for each record within the database.

Plant

Global_ID	Plant_ID	Plant Name	Plant Type	Bidder_ID
1	1	Stuyvesant	Hopper	235
2	2	Weeks	Hopper	622
3	3	Brunswick	Pipeline	24

235

Global_ID	JobKey	Bidder No.	Bid Price	Winning Bidder
4	17SAS001	1	12690124	True
5	17SAM014	3	17506523	False

Bidder

Global_ID	Bidder_ID	Bidder Name
6	235	Dutra Dredging Co.
7	622	Weeks Marine, Inc (ATLANTIC)
8	24	Southern Dredging Co., Inc.

Ref: https://www.digitalocean.com/community/tutorials/understanding-relational-databases

ACID Database Concept

- **Atomicity** – Transactions succeed or fail completely.
- **Consistency** – Database can go only from one valid state to another.
- **Isolation** – Transactions happen independently.
- **Durability** – Records persist in in non-volatile memory.

RDBMS Pros?

- Rigid schema.
- Structured nature makes it easily sortable and searchable.
- Highly flexible relational design.
- Prevalent, mature technology.
- Transactional.
- Strong data integrity.

Name	Age	Education	Occupation	Likes	Breed	Tricks
Brandan	41	PhD	Engineer	-	-	-
Slater	4	-	-	Lego	-	-
Ginger	4	-	-	-	Hound	Sit

RDBMS Cons?

- Poor horizontal (distributed) scaling.
  - Writing to multiple nodes increases transaction latency and potential for corruption.
- Inefficient for unstructured data (non-tabular).

What is a NoSQL Database?

- SQL RDBMS systems impose a high level of structure on managed data.
- A non-structured (NoSQL) database provides a higher level of flexibility in the way data are stored.
  - Document
  - Columnar
  - Key: Value
  - Graph

Document Stores

- The database stores individual records as “documents”
  - JSON
  - BSON
  - XML
- Documents are independent: no foreign key requirement.
- Schema is flexible and not necessarily uniform.
- Not ACID

```
{name: ‘Slater’,      {name: ‘Brandan’,            {name: ‘Ginger’,
   age: 4,               age: 41,                     age: 4,
   likes: ‘Lego’}        occupation: ‘engineer’,      breed: ‘hound’,
                         education: ‘PhD’}            tricks: ‘sit’}
```

Columnar

- Store data as a set of columns with rows.
- Schema is flexible and not necessarily uniform.
- Pros: Scalable, responsive, compressible.
- Cons: Inefficient for online processing, data loading, and row-specific queries.

Slater	Brandan,	Ginger
Age: 4	Age: 41	Age: 4
Likes: ‘Lego’	Occupation: ‘engineer’	Breed: ‘hound’
	Education: ‘PhD’	Tricks: ‘sit’

Key: Value

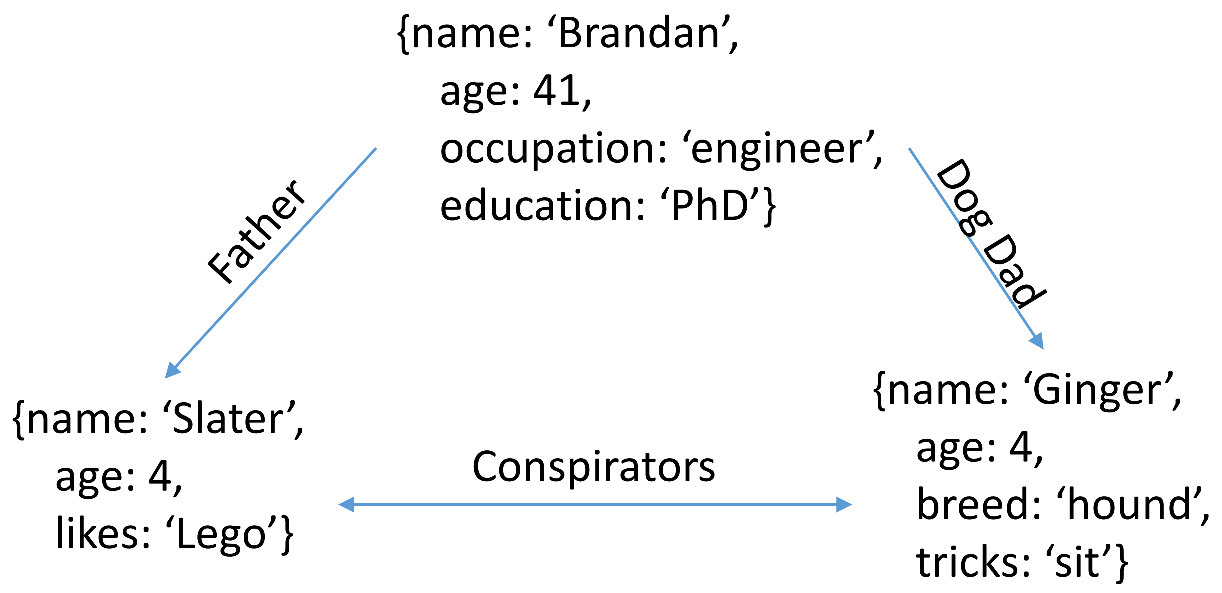
- Essentially, a dictionary.
- Pros:
  - Simple.
  - Fast.
  - Scalable.
- Cons:
  - Parser required for multiple values.
  - Not optimized for lookup.

```
Age = { 'Brandan': 41,
        'Slater': 4,
        'Ginger': 4}
```

```
{ 'Brandan': {age : 41,
              occupation: engineer},
  'Slater': {age: 4,
              likes: Lego},
  'Ginger': {age: 4,
              tricks: sit}
}
```

Graph

- A collection of nodes and edges where nodes represent entities and edges represent relationships between entities.
  - Optimized for understanding the relationship between document-type records.
- Pros:
  - Object oriented
  - Index-free adjacency
- Cons:
  - Not transaction-based



Pandas Data Functions

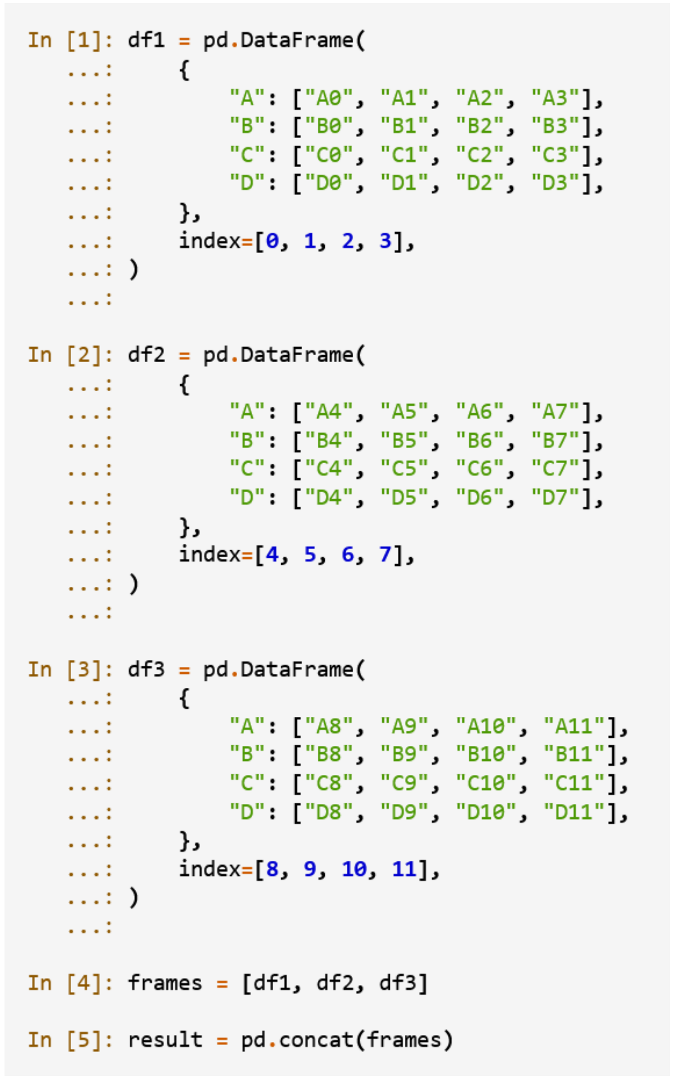
- Concatenate
- Merge
- Join

Pandas Concat

```
pd.concat(
    objs,
    axis=0,
    join="outer",
    ignore_index=False,
    keys=None,
    levels=None,
    names=None,
    verify_integrity=False,
    copy=True,
)
```

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

Pandas Concat



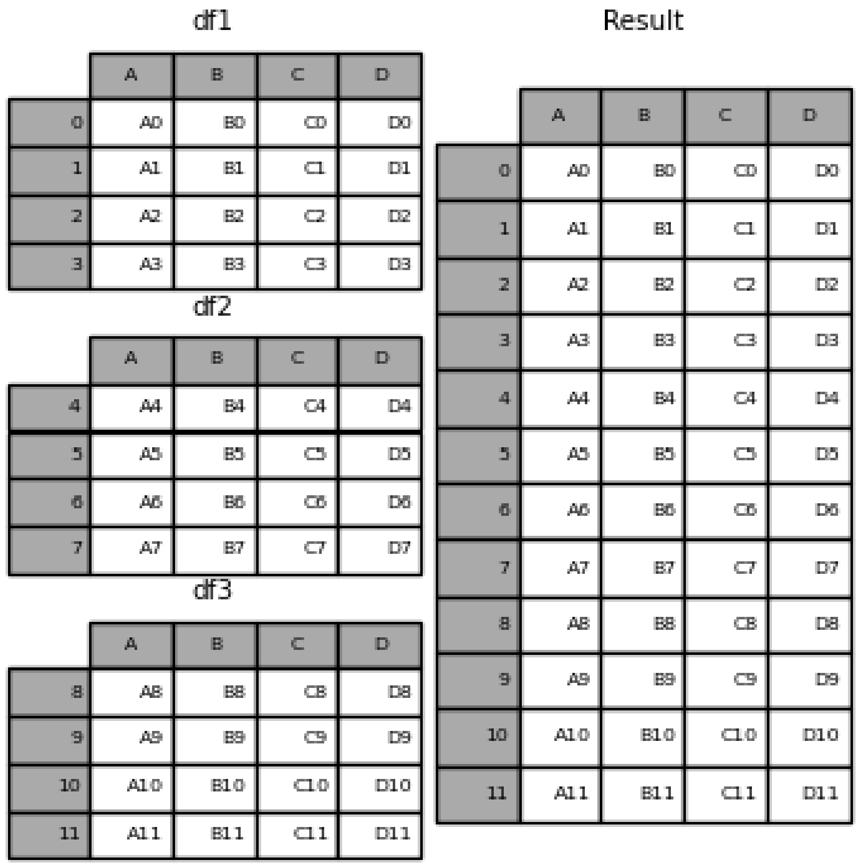
```
In [1]: df1 = pd.DataFrame(
   ...:     {
   ...:         "A": ["A0", "A1", "A2", "A3"],
   ...:         "B": ["B0", "B1", "B2", "B3"],
   ...:         "C": ["C0", "C1", "C2", "C3"],
   ...:         "D": ["D0", "D1", "D2", "D3"],
   ...:     },
   ...:     index=[0, 1, 2, 3],
   ...: )
   ...:

In [2]: df2 = pd.DataFrame(
   ...:     {
   ...:         "A": ["A4", "A5", "A6", "A7"],
   ...:         "B": ["B4", "B5", "B6", "B7"],
   ...:         "C": ["C4", "C5", "C6", "C7"],
   ...:         "D": ["D4", "D5", "D6", "D7"],
   ...:     },
   ...:     index=[4, 5, 6, 7],
   ...: )
   ...:

In [3]: df3 = pd.DataFrame(
   ...:     {
   ...:         "A": ["A8", "A9", "A10", "A11"],
   ...:         "B": ["B8", "B9", "B10", "B11"],
   ...:         "C": ["C8", "C9", "C10", "C11"],
   ...:         "D": ["D8", "D9", "D10", "D11"],
   ...:     },
   ...:     index=[8, 9, 10, 11],
   ...: )
   ...:

In [4]: frames = [df1, df2, df3]

In [5]: result = pd.concat(frames)
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

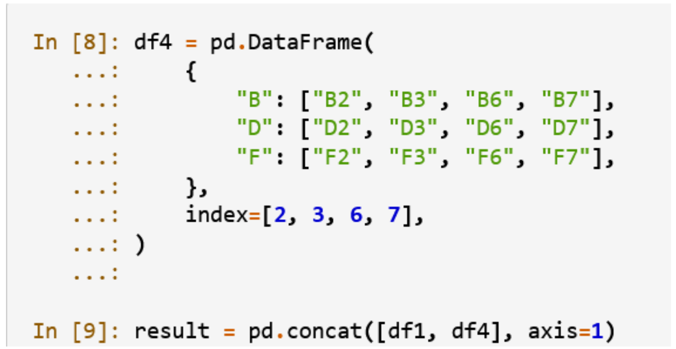
Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

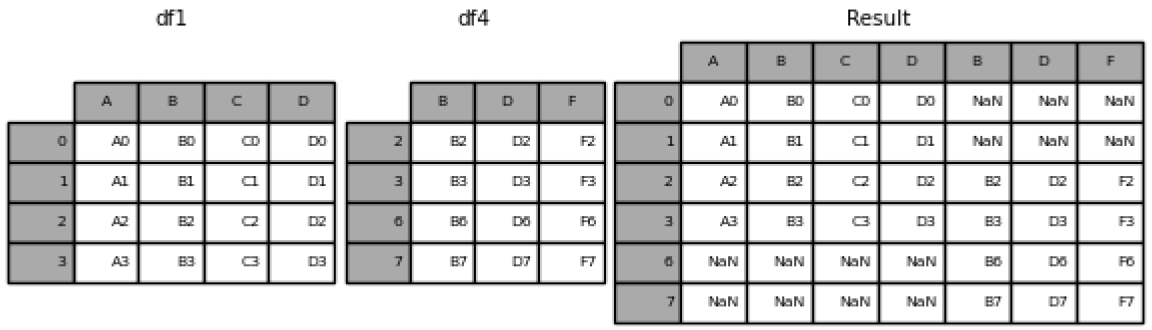
Pandas Concat: join = “outer” vs. “inner”

Join = ‘outer’ (default)



```
In [8]: df4 = pd.DataFrame(
   ...:     {
   ...:         "B": ["B2", "B3", "B6", "B7"],
   ...:         "D": ["D2", "D3", "D6", "D7"],
   ...:         "F": ["F2", "F3", "F6", "F7"],
   ...:     },
   ...:     index=[2, 3, 6, 7],
   ...: )
   ...:

In [9]: result = pd.concat([df1, df4], axis=1)
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

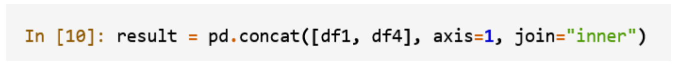
df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

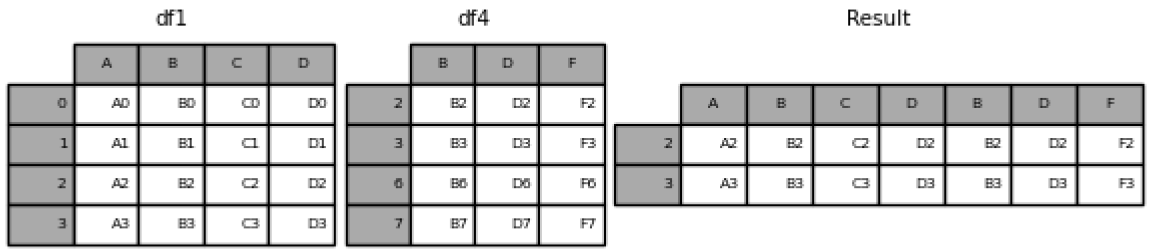
Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3
6	NaN	NaN	NaN	NaN	B6	D6	F6
7	NaN	NaN	NaN	NaN	B7	D7	F7

Join = ‘inner’



```
In [10]: result = pd.concat([df1, df4], axis=1, join="inner")
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

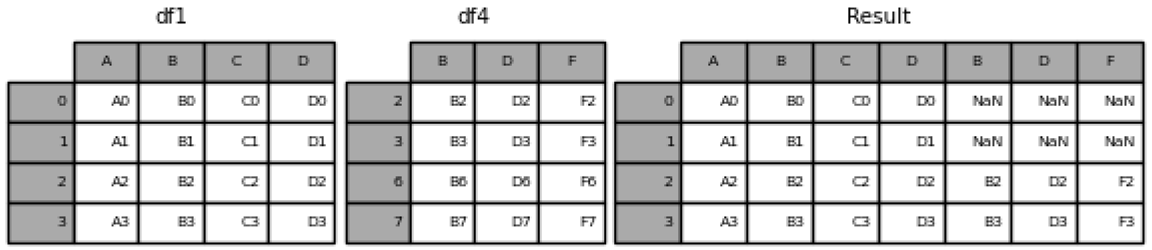
Ref: https://pandas.pydata.org/docs/user_guide/merging.html

Pandas Concat: reindexing

```
In [11]: result = pd.concat([df1, df4], axis=1).reindex(df1.index)
```

or

```
In [12]: pd.concat([df1, df4.reindex(df1.index)], axis=1)
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

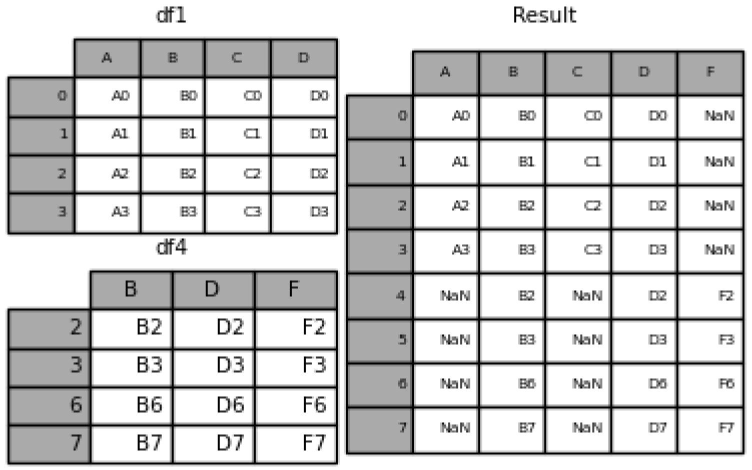
df4

	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	B	D	F
0	A0	B0	C0	D0	NaN	NaN	NaN
1	A1	B1	C1	D1	NaN	NaN	NaN
2	A2	B2	C2	D2	B2	D2	F2
3	A3	B3	C3	D3	B3	D3	F3

```
In [13]: result = pd.concat([df1, df4], ignore_index=True, sort=False)
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df4

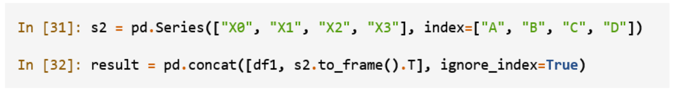
	B	D	F
2	B2	D2	F2
3	B3	D3	F3
6	B6	D6	F6
7	B7	D7	F7

Result

	A	B	C	D	F
0	A0	B0	C0	D0	NaN
1	A1	B1	C1	D1	NaN
2	A2	B2	C2	D2	NaN
3	A3	B3	C3	D3	NaN
4	NaN	B2	NaN	D2	F2
5	NaN	B3	NaN	D3	F3
6	NaN	B6	NaN	D6	F6
7	NaN	B7	NaN	D7	F7

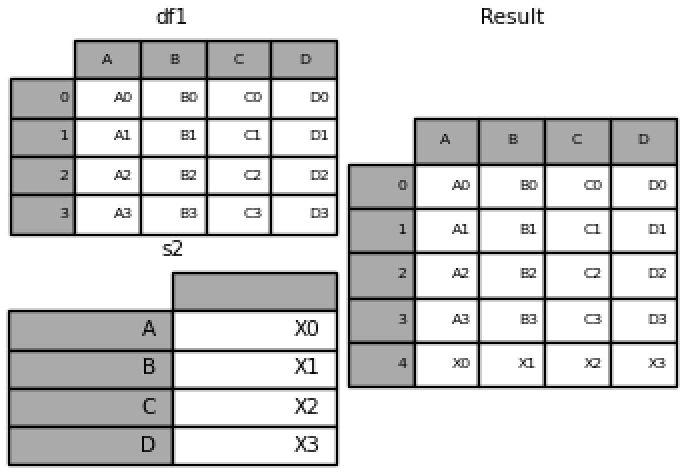
Ref: https://pandas.pydata.org/docs/user_guide/merging.html

Pandas Concat: appending



```
In [31]: s2 = pd.Series(["X0", "X1", "X2", "X3"], index=["A", "B", "C", "D"])

In [32]: result = pd.concat([df1, s2.to_frame().T], ignore_index=True)
```



df1

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

s2

A	X0
B	X1
C	X2
D	X3

Result

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	X0	X1	X2	X3

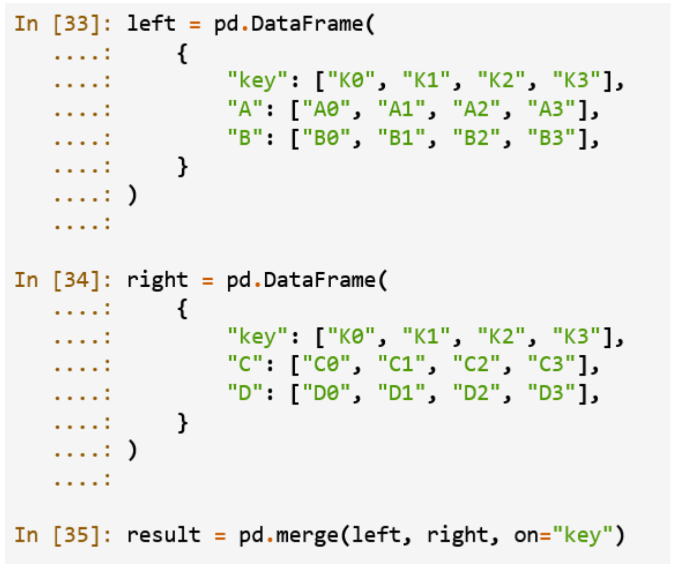
Ref: https://pandas.pydata.org/docs/user_guide/merging.html

Pandas Merge

```
pd.merge(
    left,
    right,
    how="inner",
    on=None,
    left_on=None,
    right_on=None,
    left_index=False,
    right_index=False,
    sort=True,
    suffixes=("_x", "_y"),
    copy=True,
    indicator=False,
    validate=None,
)
```

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

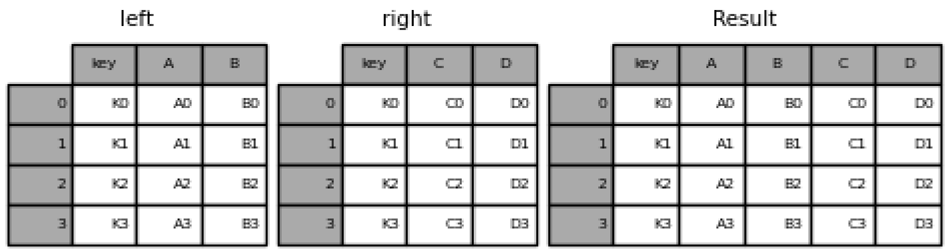
Pandas Merge



```
In [33]: left = pd.DataFrame(
   ....:     {
   ....:         "key": ["K0", "K1", "K2", "K3"],
   ....:         "A": ["A0", "A1", "A2", "A3"],
   ....:         "B": ["B0", "B1", "B2", "B3"],
   ....:     }
   ....: )
   ....:

In [34]: right = pd.DataFrame(
   ....:     {
   ....:         "key": ["K0", "K1", "K2", "K3"],
   ....:         "C": ["C0", "C1", "C2", "C3"],
   ....:         "D": ["D0", "D1", "D2", "D3"],
   ....:     }
   ....: )
   ....:

In [35]: result = pd.merge(left, right, on="key")
```



left

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

right

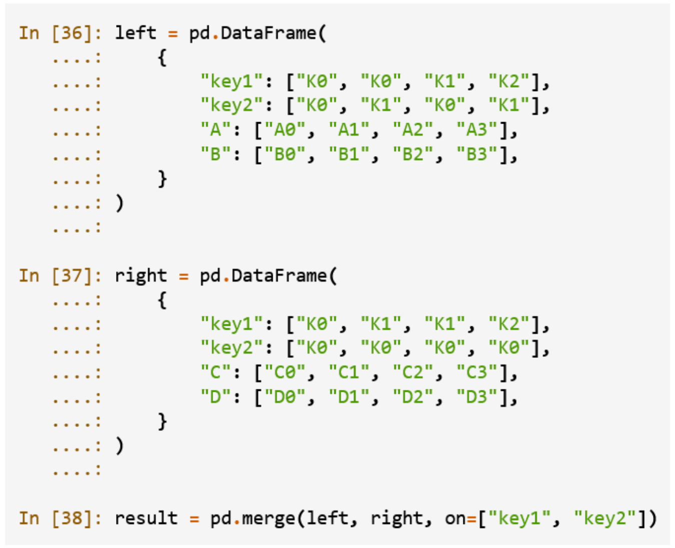
	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

Result

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

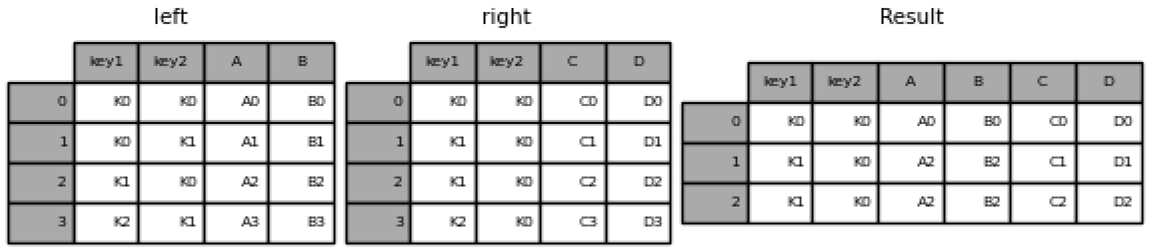
Pandas Merge



```
In [36]: left = pd.DataFrame(
   ....:     {
   ....:         "key1": ["K0", "K0", "K1", "K2"],
   ....:         "key2": ["K0", "K1", "K0", "K1"],
   ....:         "A": ["A0", "A1", "A2", "A3"],
   ....:         "B": ["B0", "B1", "B2", "B3"],
   ....:     }
   ....: )
   ....:

In [37]: right = pd.DataFrame(
   ....:     {
   ....:         "key1": ["K0", "K1", "K1", "K2"],
   ....:         "key2": ["K0", "K0", "K0", "K0"],
   ....:         "C": ["C0", "C1", "C2", "C3"],
   ....:         "D": ["D0", "D1", "D2", "D3"],
   ....:     }
   ....: )
   ....:

In [38]: result = pd.merge(left, right, on=["key1", "key2"])
```



left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

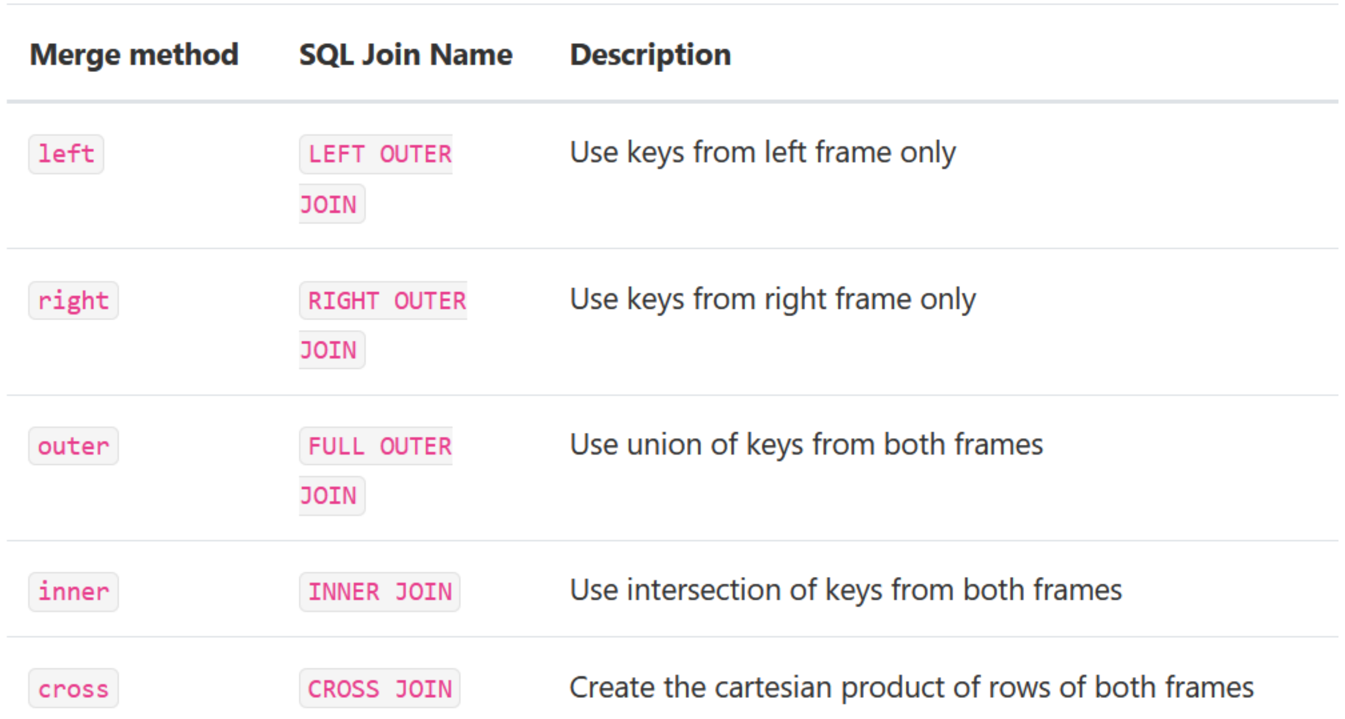
	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

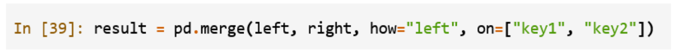
Pandas Merge: How options



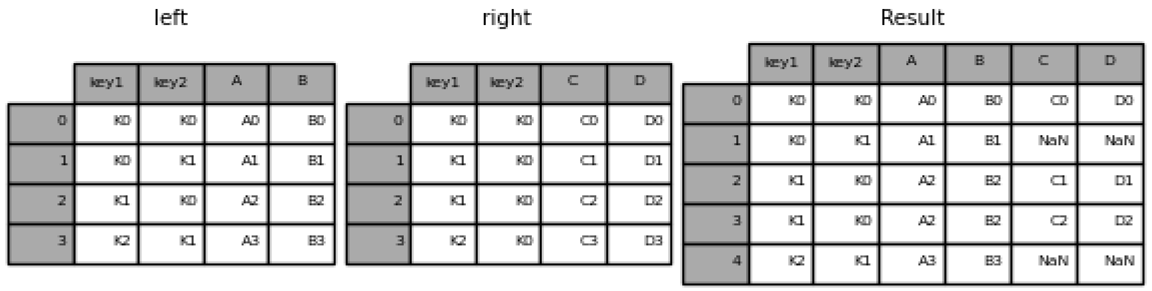
Merge method	SQL Join Name	Description
`left`	`LEFT OUTER JOIN`	Use keys from left frame only
`right`	`RIGHT OUTER JOIN`	Use keys from right frame only
`outer`	`FULL OUTER JOIN`	Use union of keys from both frames
`inner`	`INNER JOIN`	Use intersection of keys from both frames
`cross`	`CROSS JOIN`	Create the cartesian product of rows of both frames

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

Pandas Merge: How options: “left” vs. “right”



```
In [39]: result = pd.merge(left, right, how="left", on=["key1", "key2"])
```



left

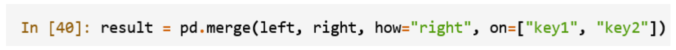
	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

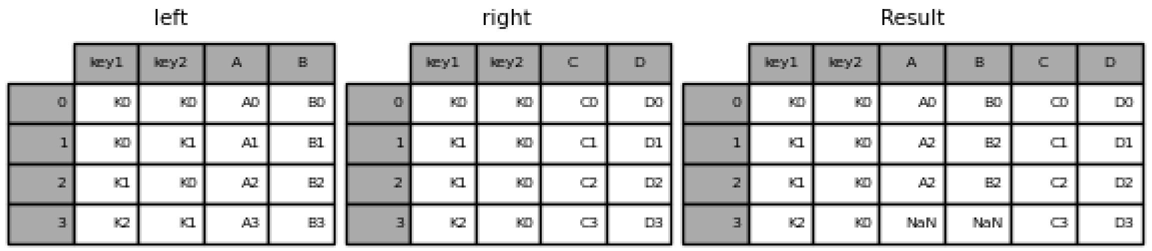
	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN



```
In [40]: result = pd.merge(left, right, how="right", on=["key1", "key2"])
```



left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

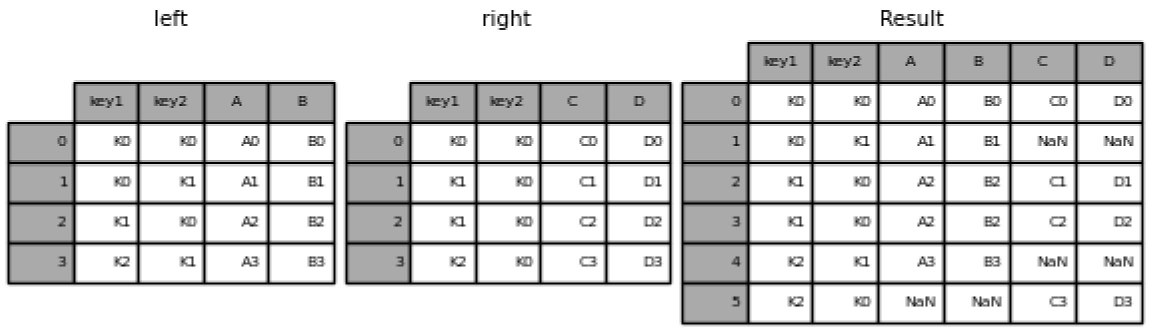
Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2
3	K2	K0	NaN	NaN	C3	D3

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

Pandas Merge: How options: “outer” vs. “inner”

```
In [41]: result = pd.merge(left, right, how="outer", on=["key1", "key2"])
```



left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

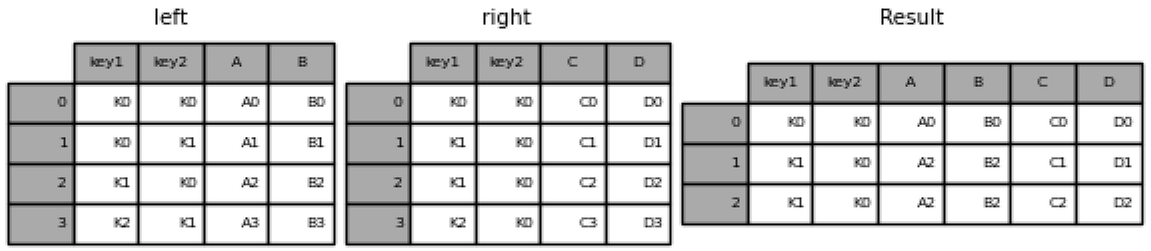
right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K0	K1	A1	B1	NaN	NaN
2	K1	K0	A2	B2	C1	D1
3	K1	K0	A2	B2	C2	D2
4	K2	K1	A3	B3	NaN	NaN
5	K2	K0	NaN	NaN	C3	D3

```
In [42]: result = pd.merge(left, right, how="inner", on=["key1", "key2"])
```



left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

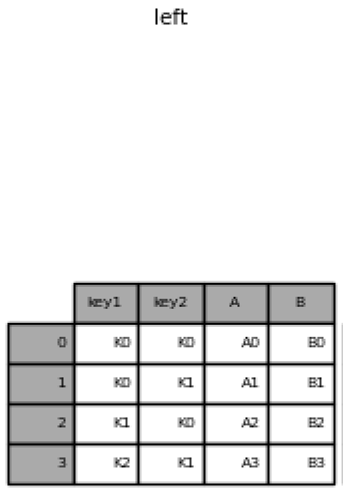
Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

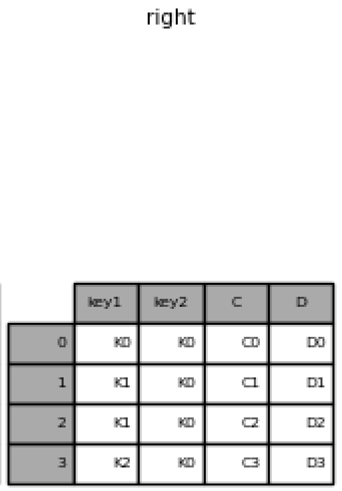
Pandas Merge: How options: “cross”

```
In [43]: result = pd.merge(left, right, how="cross")
```



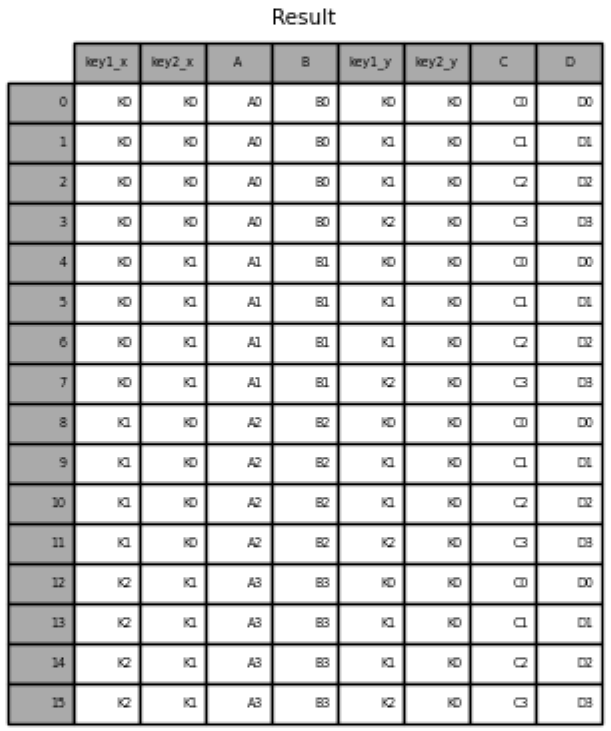
left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3



right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3



Result

	key1_x	key2_x	A	B	key1_y	key2_y	C	D
0	K0	K0	A0	B0	K0	K0	C0	D0
1	K0	K0	A0	B0	K1	K0	C1	D1
2	K0	K0	A0	B0	K1	K0	C2	D2
3	K0	K0	A0	B0	K2	K0	C3	D3
4	K0	K1	A1	B1	K0	K0	C0	D0
5	K0	K1	A1	B1	K1	K0	C1	D1
6	K0	K1	A1	B1	K1	K0	C2	D2
7	K0	K1	A1	B1	K2	K0	C3	D3
8	K1	K0	A2	B2	K0	K0	C0	D0
9	K1	K0	A2	B2	K1	K0	C1	D1
10	K1	K0	A2	B2	K1	K0	C2	D2
11	K1	K0	A2	B2	K2	K0	C3	D3
12	K2	K1	A3	B3	K0	K0	C0	D0
13	K2	K1	A3	B3	K1	K0	C1	D1
14	K2	K1	A3	B3	K1	K0	C2	D2
15	K2	K1	A3	B3	K2	K0	C3	D3

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

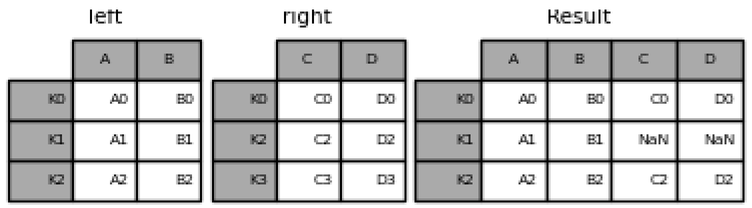
Pandas Join: on index

how = ‘left’ (default)

```
In [79]: left = pd.DataFrame(
   ....:     {"A": ["A0", "A1", "A2"], "B": ["B0", "B1", "B2"]}, index=["K0", "K1", "K2"]
   ....: )
   ....:

In [80]: right = pd.DataFrame(
   ....:     {"C": ["C0", "C2", "C3"], "D": ["D0", "D2", "D3"]}, index=["K0", "K2", "K3"]
   ....: )
   ....:

In [81]: result = left.join(right)
```


left
right
Result

left

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

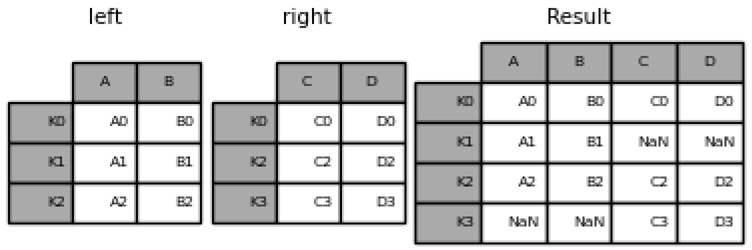
right

	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

Result

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2

```
In [82]: result = left.join(right, how="outer")
```


left
right
Result

left

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

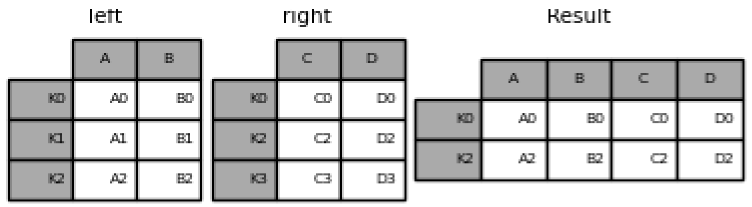
right

	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

Result

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3

```
In [83]: result = left.join(right, how="inner")
```


left
right
Result

left

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

right

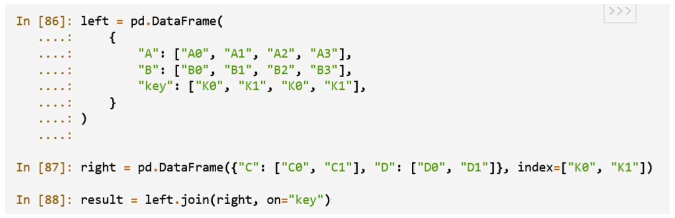
	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

Result

	A	B	C	D
K0	A0	B0	C0	D0
K2	A2	B2	C2	D2

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

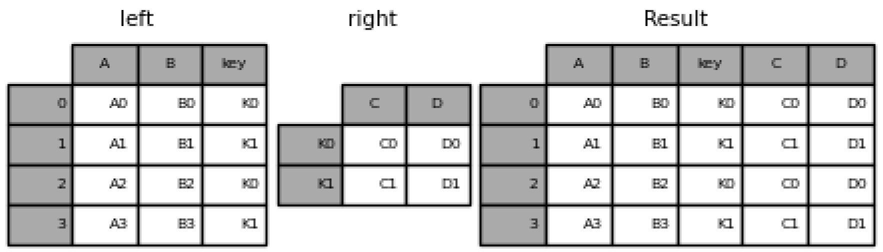
Pandas Join: key columns on index



```
In [86]: left = pd.DataFrame(
   ....:     {
   ....:         "A": ["A0", "A1", "A2", "A3"],
   ....:         "B": ["B0", "B1", "B2", "B3"],
   ....:         "key": ["K0", "K1", "K0", "K1"],
   ....:     }
   ....: )
   ....:

In [87]: right = pd.DataFrame({"C": ["C0", "C1"], "D": ["D0", "D1"]}, index=["K0", "K1"])

In [88]: result = left.join(right, on="key")
```



left

	A	B	key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K0
3	A3	B3	K1

right

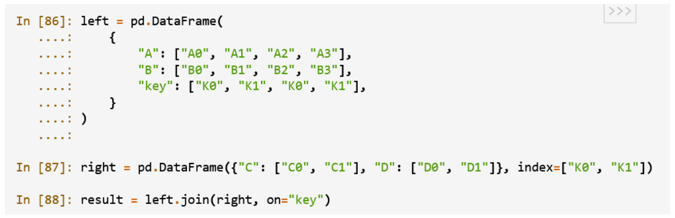
	C	D
K0	C0	D0
K1	C1	D1

Result

	A	B	key	C	D
0	A0	B0	K0	C0	D0
1	A1	B1	K1	C1	D1
2	A2	B2	K0	C0	D0
3	A3	B3	K1	C1	D1

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

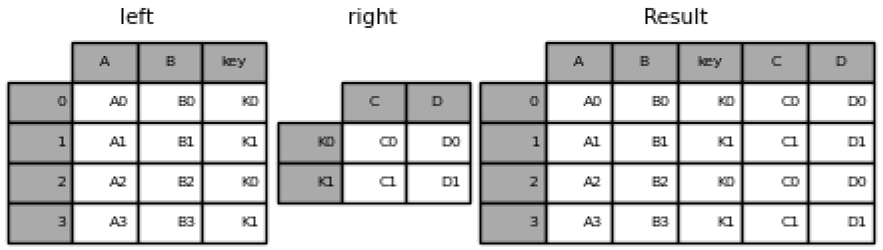
Pandas Compare: key columns on index



```
In [86]: left = pd.DataFrame(
   ....:     {
   ....:         "A": ["A0", "A1", "A2", "A3"],
   ....:         "B": ["B0", "B1", "B2", "B3"],
   ....:         "key": ["K0", "K1", "K0", "K1"],
   ....:     }
   ....: )
   ....:

In [87]: right = pd.DataFrame({"C": ["C0", "C1"], "D": ["D0", "D1"]}, index=["K0", "K1"])

In [88]: result = left.join(right, on="key")
```



left

	A	B	key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K0
3	A3	B3	K1

right

	C	D
K0	C0	D0
K1	C1	D1

Result

	A	B	key	C	D
0	A0	B0	K0	C0	D0
1	A1	B1	K1	C1	D1
2	A2	B2	K0	C0	D0
3	A3	B3	K1	C1	D1

Ref: https://pandas.pydata.org/docs/user_guide/merging.html

Data Cleaning: A Love Story

Use Case

The USACE maintains two stovepiped databases:

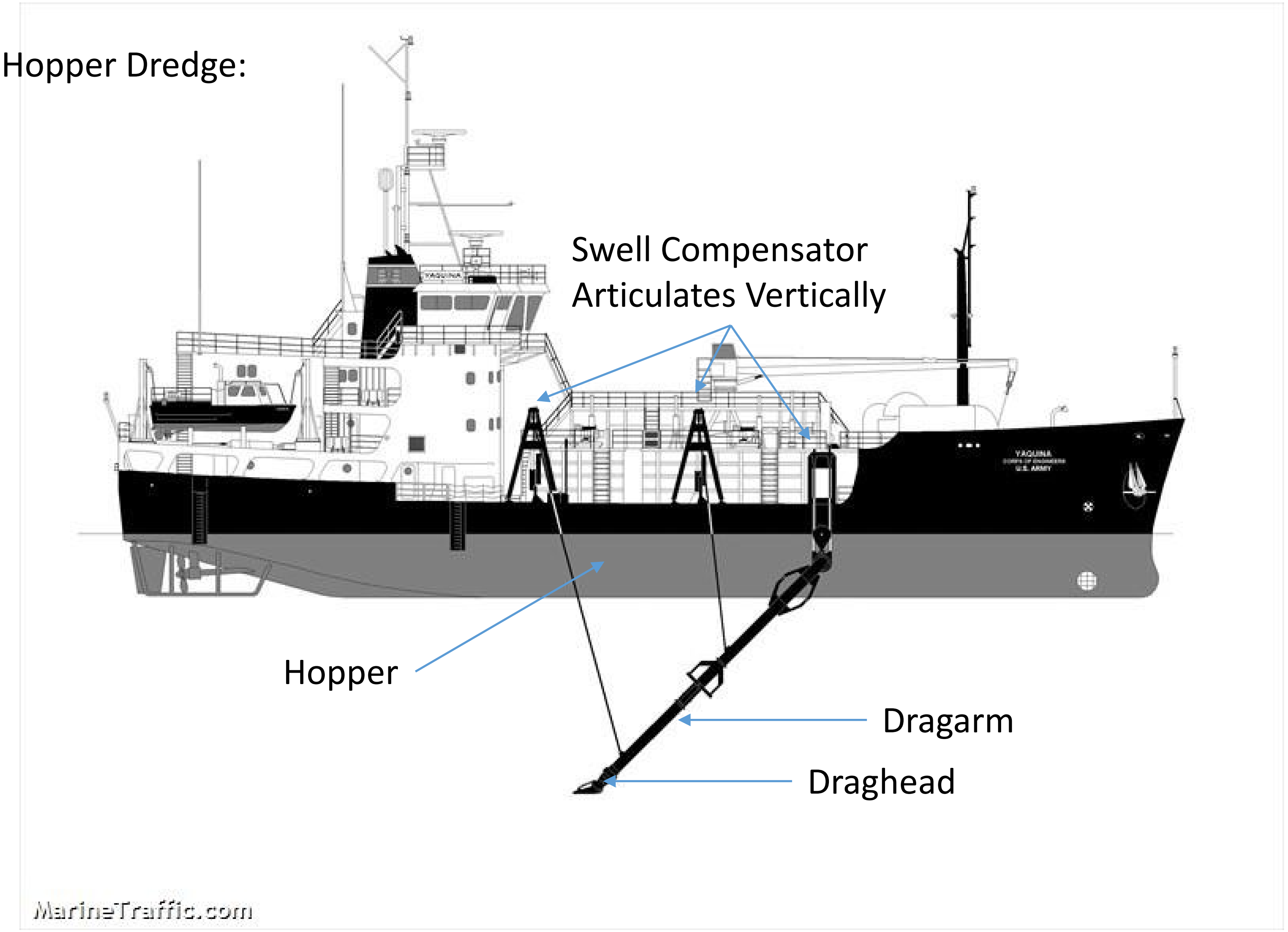
The Dredge Quality Management (DQM) database tracks the operational aspects of the dredge fleet.

The Dredging Information System (DIS) database tracks the administrative aspects of the dredging program.

GOAL: Figure out the data inconsistencies that need to be remedied to get them talking to each other to develop and leverage operational intelligence.

What is a dredge?

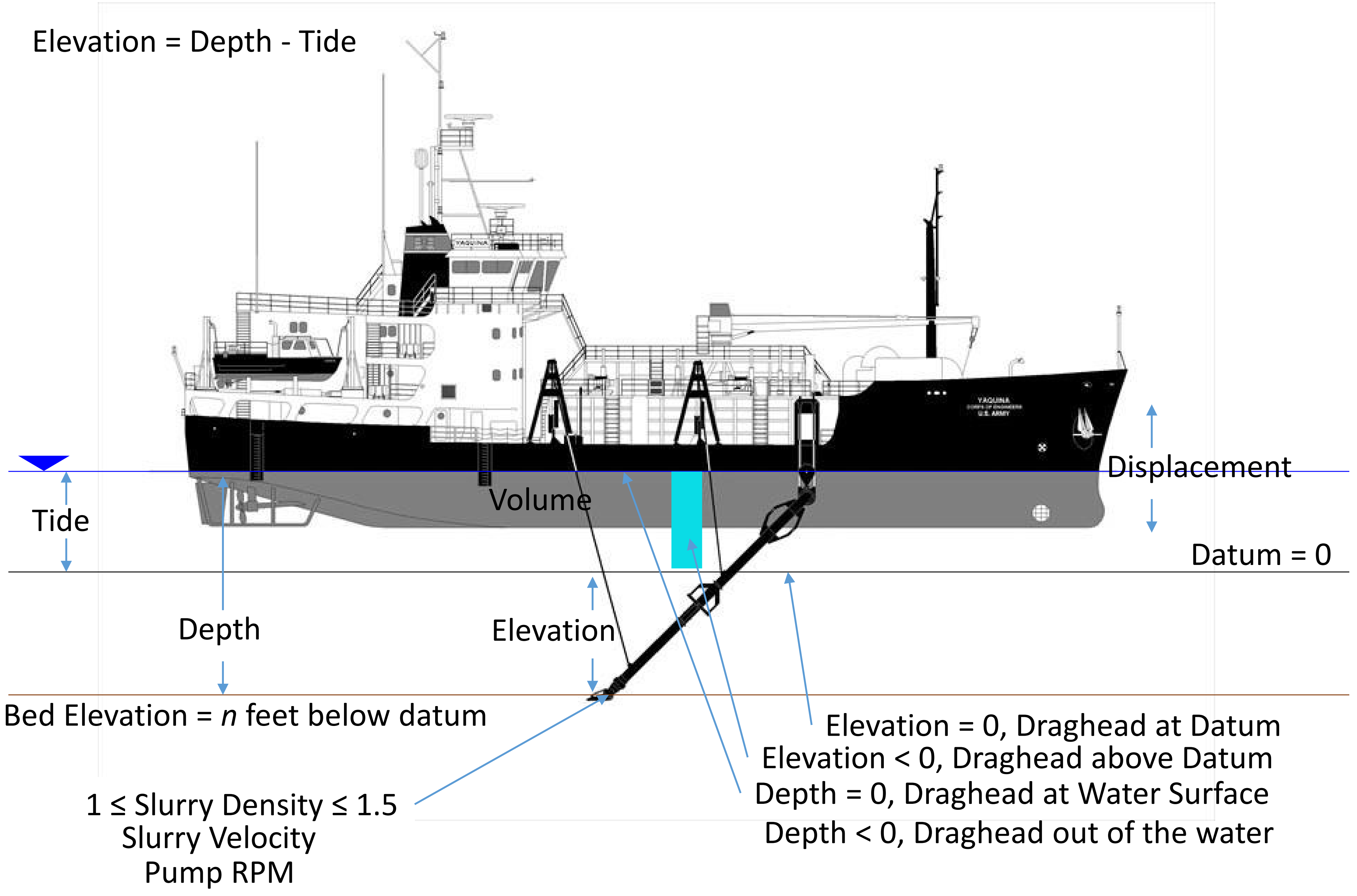
Trailing Suction Hopper Dredge:
YAQUINA



What’s the point of DQM?

- Monitoring dredge operations on dredging projects
  - Dredging in the right spot
  - Dumping in the right spot
- In addition to Trailing Suction Hopper Dredges DQM monitors:
  - Scows (TSHD without dredging function)
  - Cutter Suction Pipeline Dredges

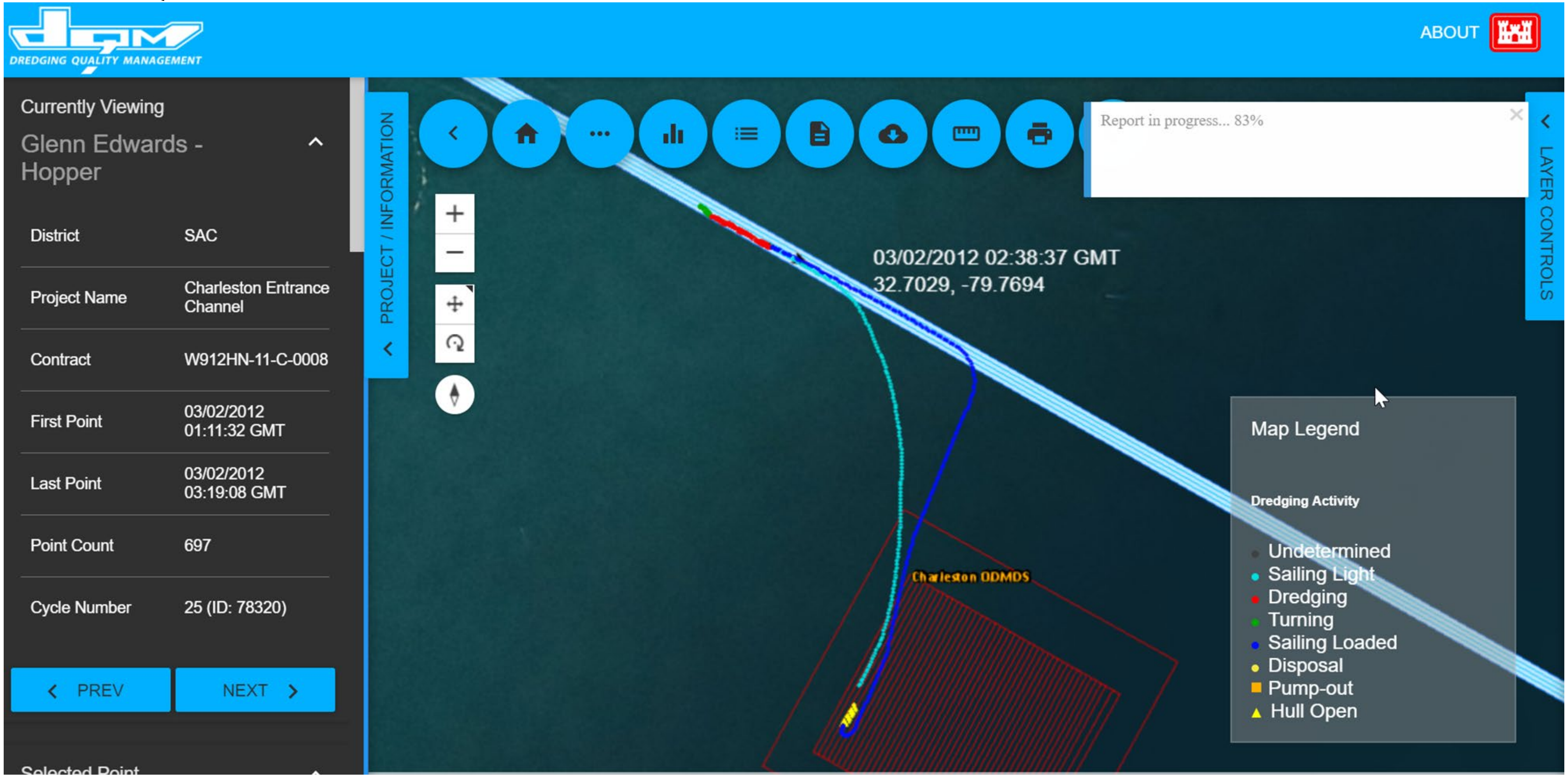
Measurements


Elevation = Depth - Tide
Displacement
Volume
Tide
Datum = 0
Depth
Elevation
Bed Elevation = *n* feet below datum
1 ≤ Slurry Density ≤ 1.5
Slurry Velocity
Pump RPM
Elevation = 0, Draghead at Datum
Elevation < 0, Draghead above Datum
Depth = 0, Draghead at Water Surface
Depth < 0, Draghead out of the water
YAQUINA
U.S. ARMY

DQM Fundaments

- DQM mandate is to monitor Active Dredging
  - Dredges do stuff outside the dredge cycle (e.g. mobilization)
- “DQM database” is >120m records
  - 2009 – 2016 (test set, hopper only)
  - ~20 unique plant
  - ~300 unique contracts
- Timing ~11s between sequential reports
- Key indices: Contract, Project, Plant, Cycle, Time, Dredge State

DQM Interface



Currently Viewing

Glenn Edwards - Hopper

District	SAC
Project Name	Charleston Entrance Channel
Contract	W912HN-11-C-0008
First Point	03/02/2012 01:11:32 GMT
Last Point	03/02/2012 03:19:08 GMT
Point Count	697
Cycle Number	25 (ID: 78320)

Report in progress... 83%

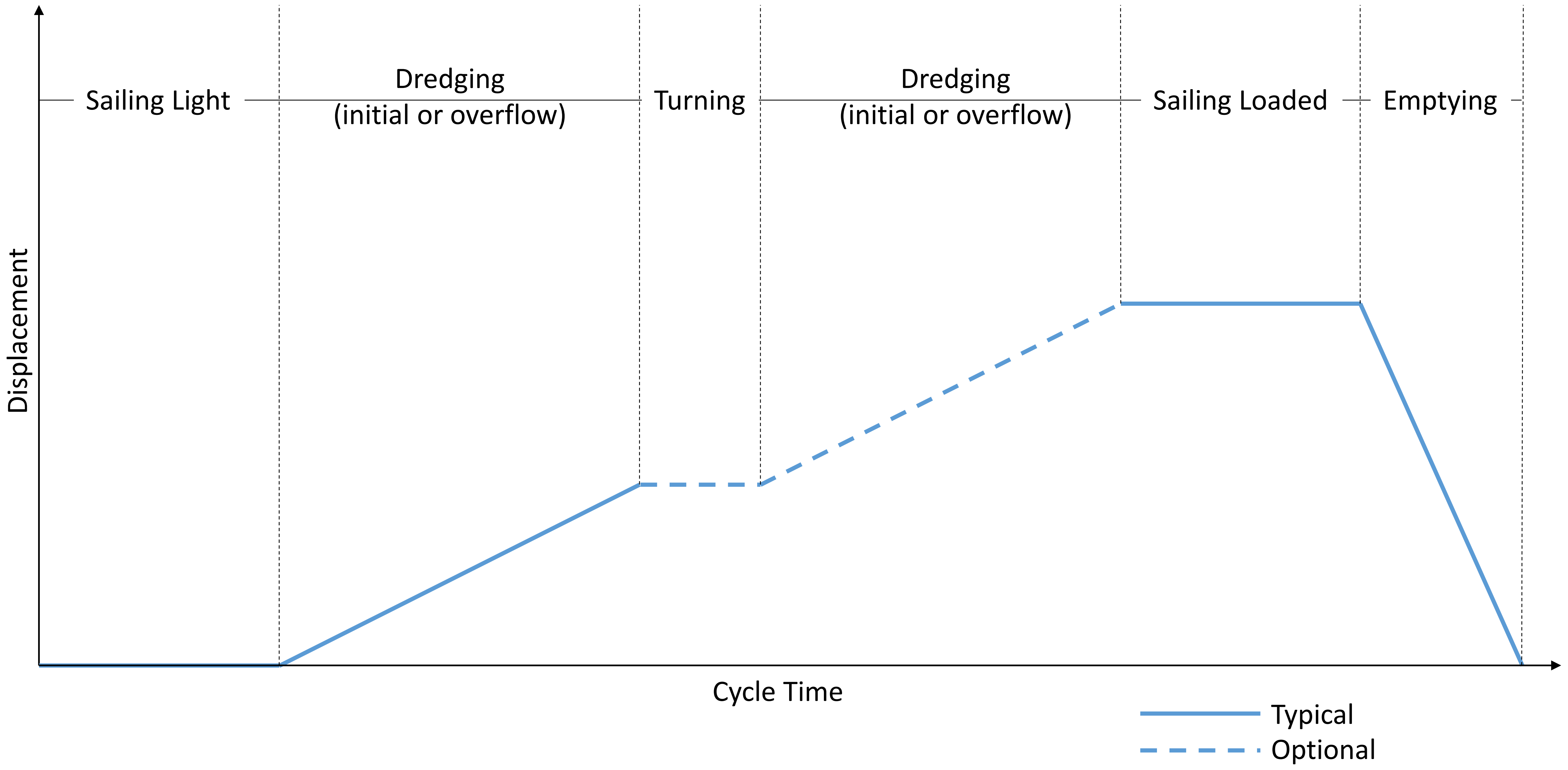
03/02/2012 02:38:37 GMT
32.7029, -79.7694

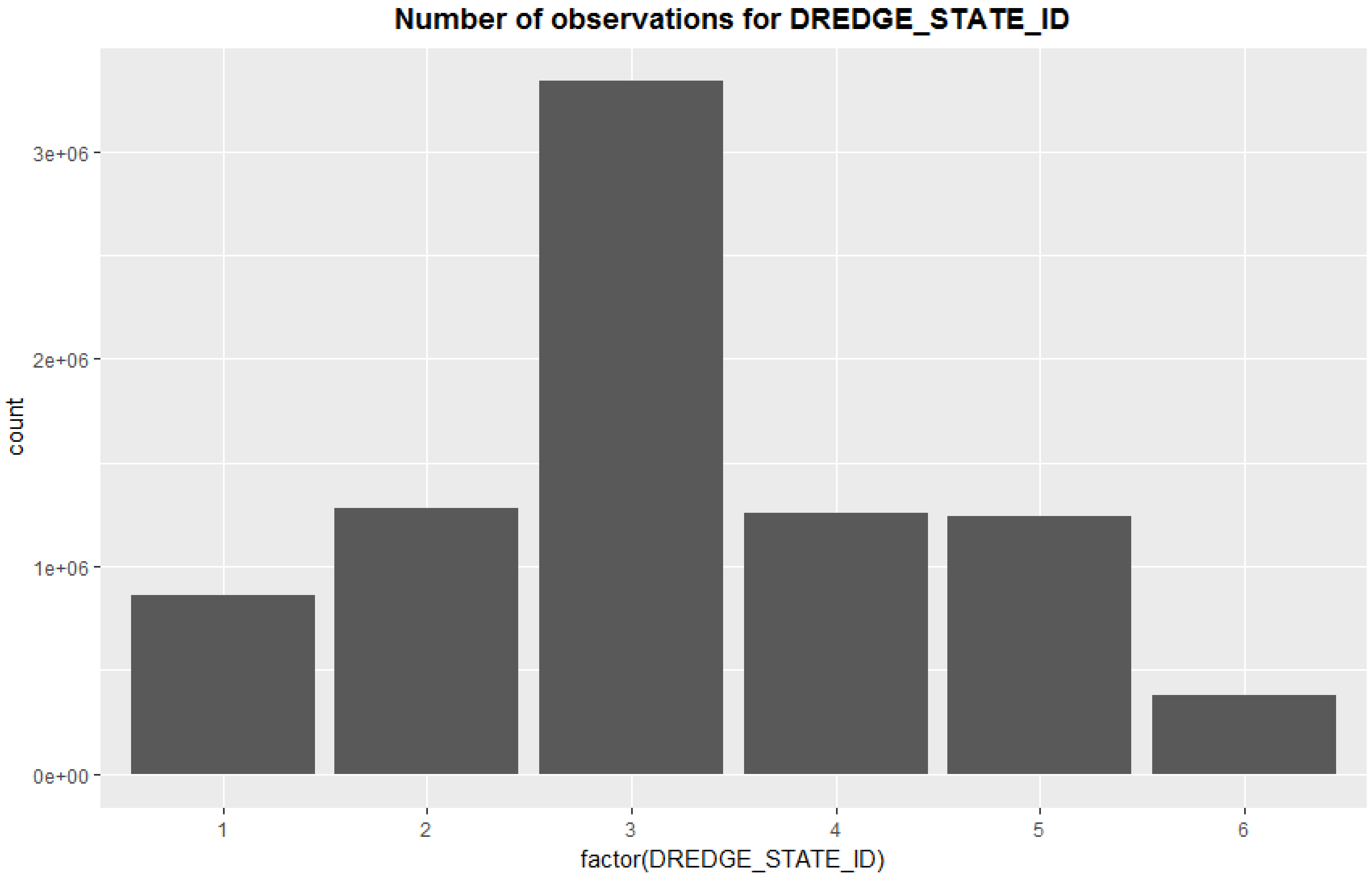
Map Legend

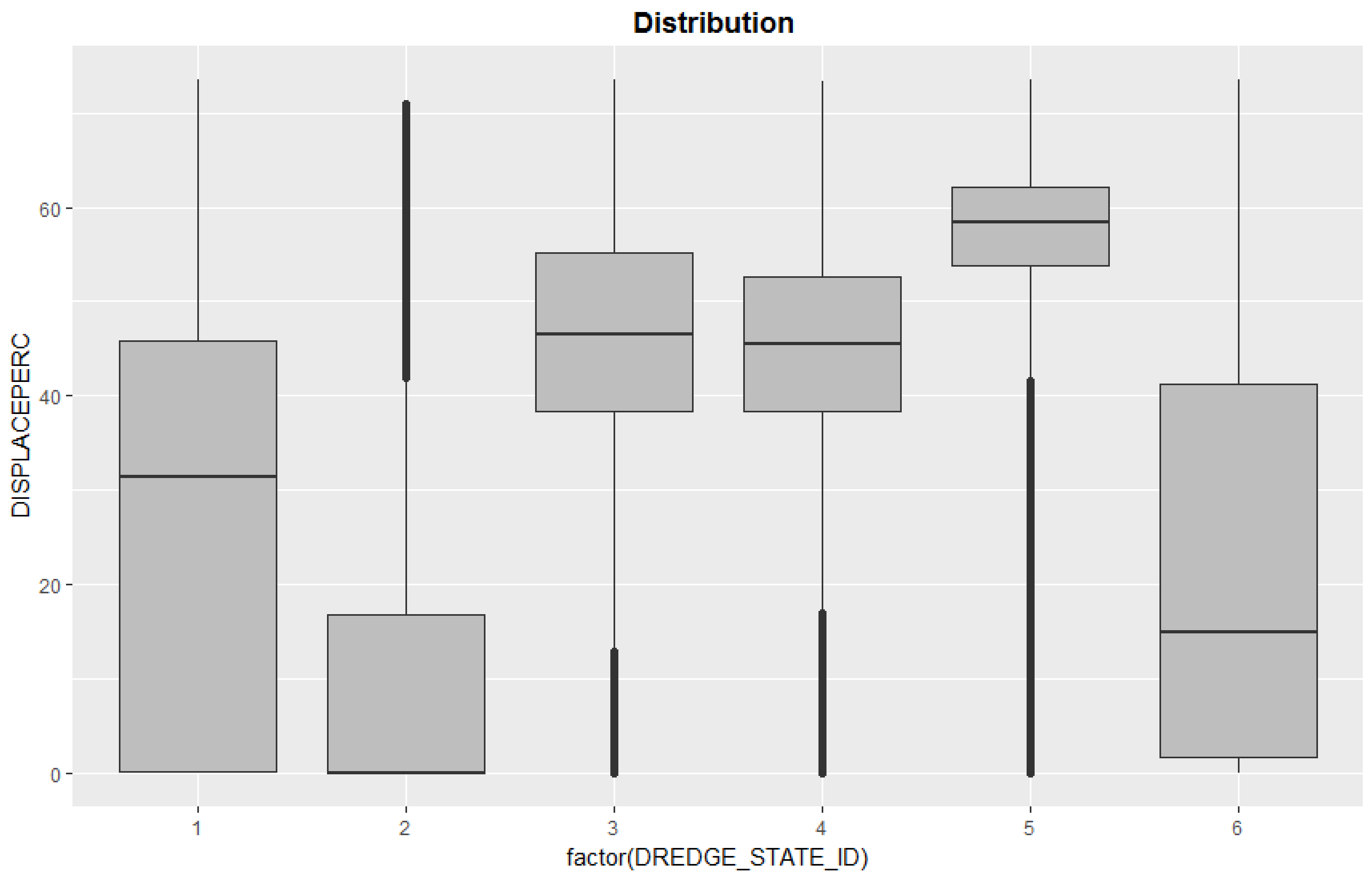
Dredging Activity

- ● Undetermined
- ● Sailing Light
- ● Dredging
- ● Turning
- ● Sailing Loaded
- ● Disposal
- ■ Pump-out
- ▲ Hull Open

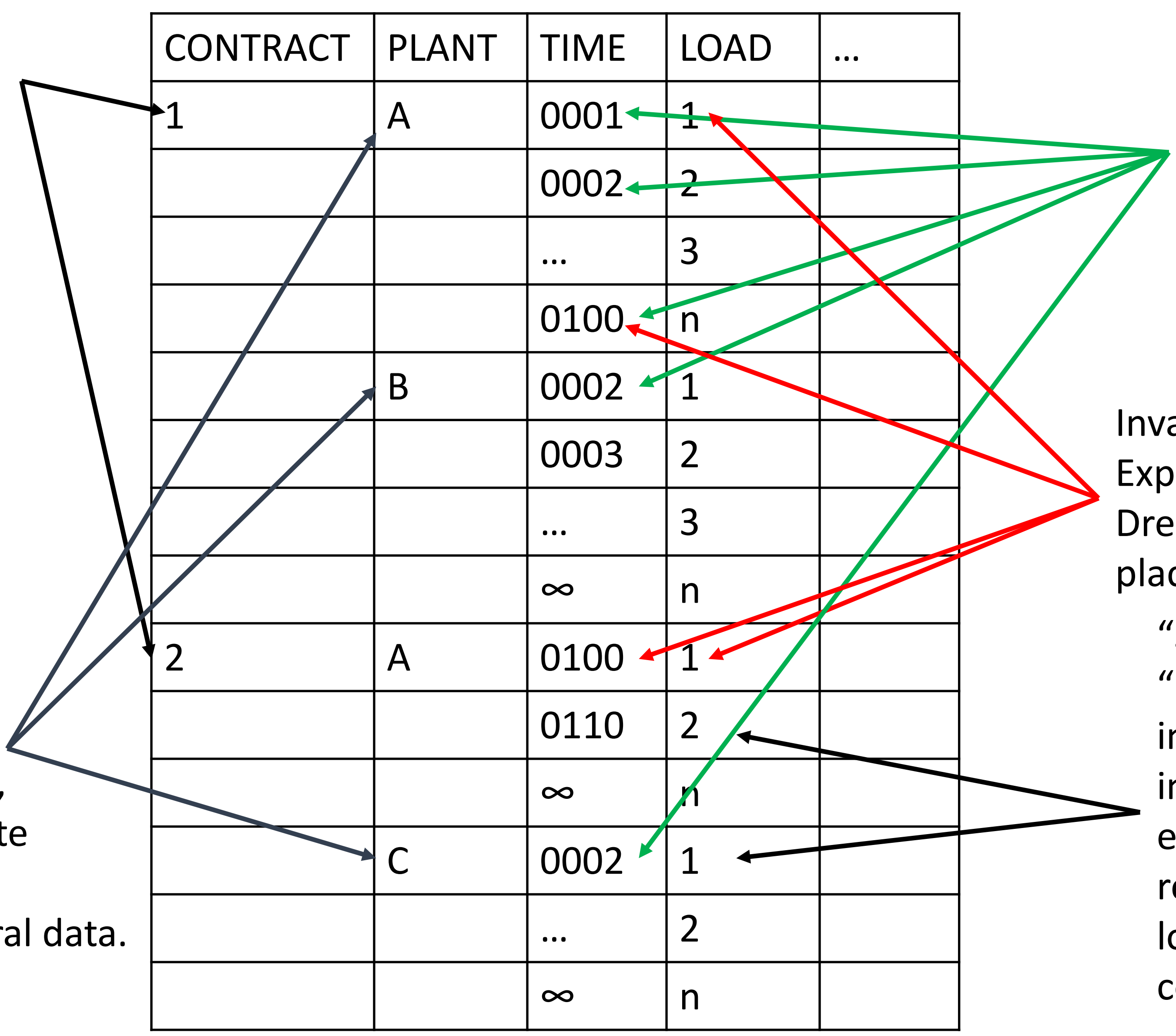
Dredge Cycle




Number of observations for DREDGE_STATE_ID
count
3e+06
2e+06
1e+06
0e+00
1
2
3
4
5
6
factor(DREDGE_STATE_ID)


Distribution
DISPLACEPERC
0
20
40
60
1
2
3
4
5
6
factor(DREDGE_STATE_ID)

DQM Record Consistency Expectation



CONTRACT	PLANT	TIME	LOAD	…
1	A	0001	1	
		0002	2	
		…	3	
		0100	n	
	B	0002	1	
		0003	2	
		…	3	
		∞	n	
2	A	0100	1	
		0110	2	
		∞	n	
	C	0002	1	
		…	2	
		∞	n	

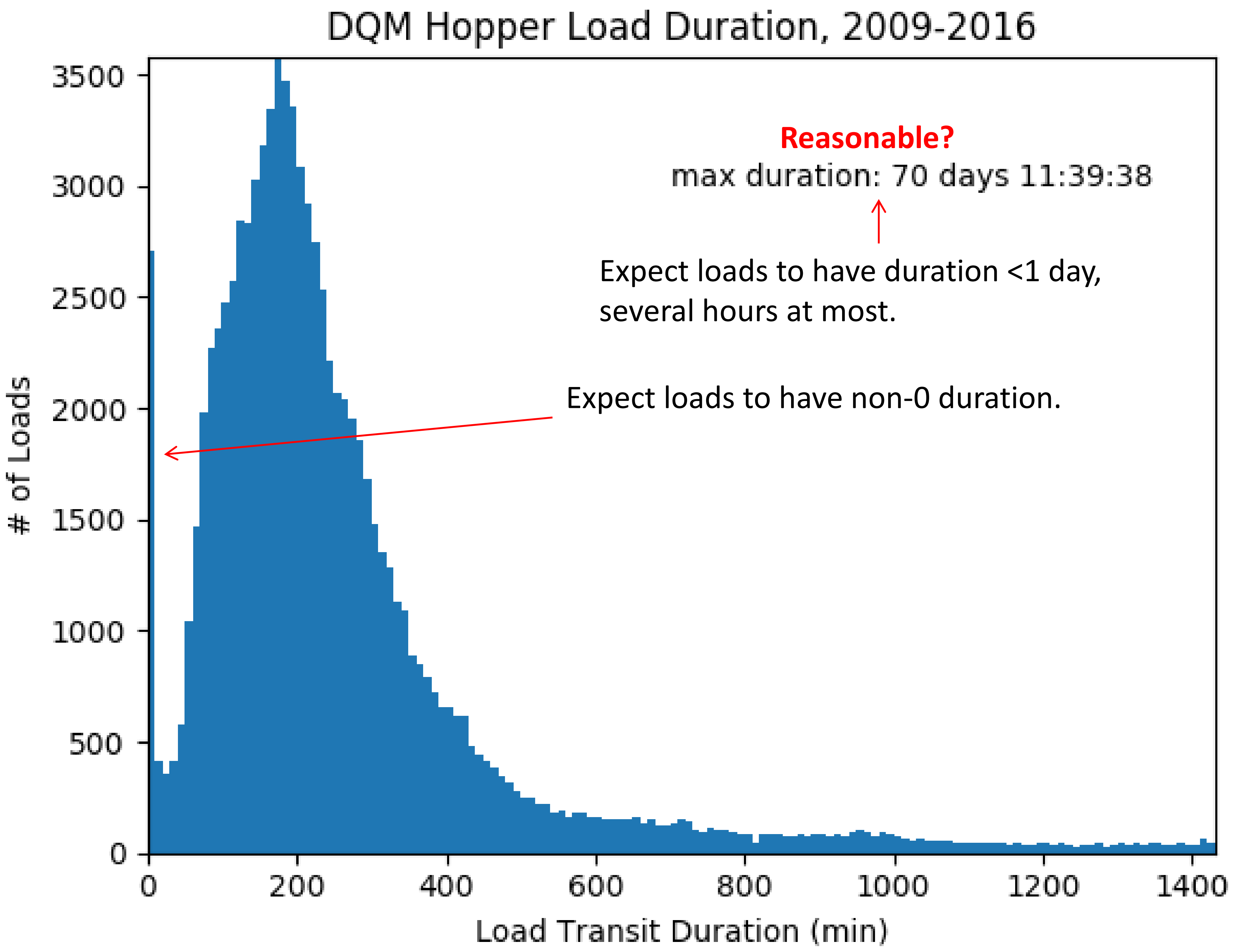
‘unique’, with spatio-temporal components

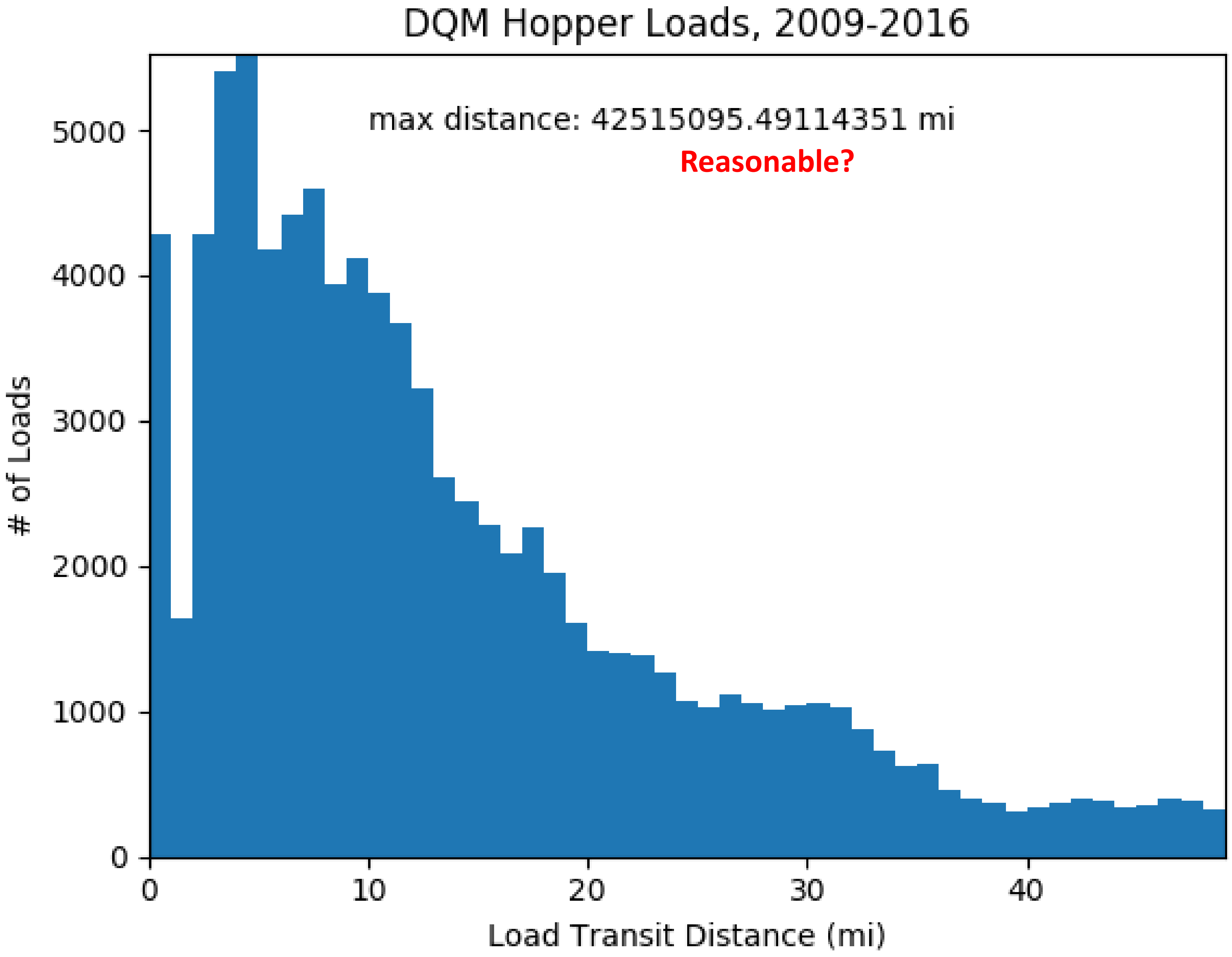
unique, generate spatio-temporal data.

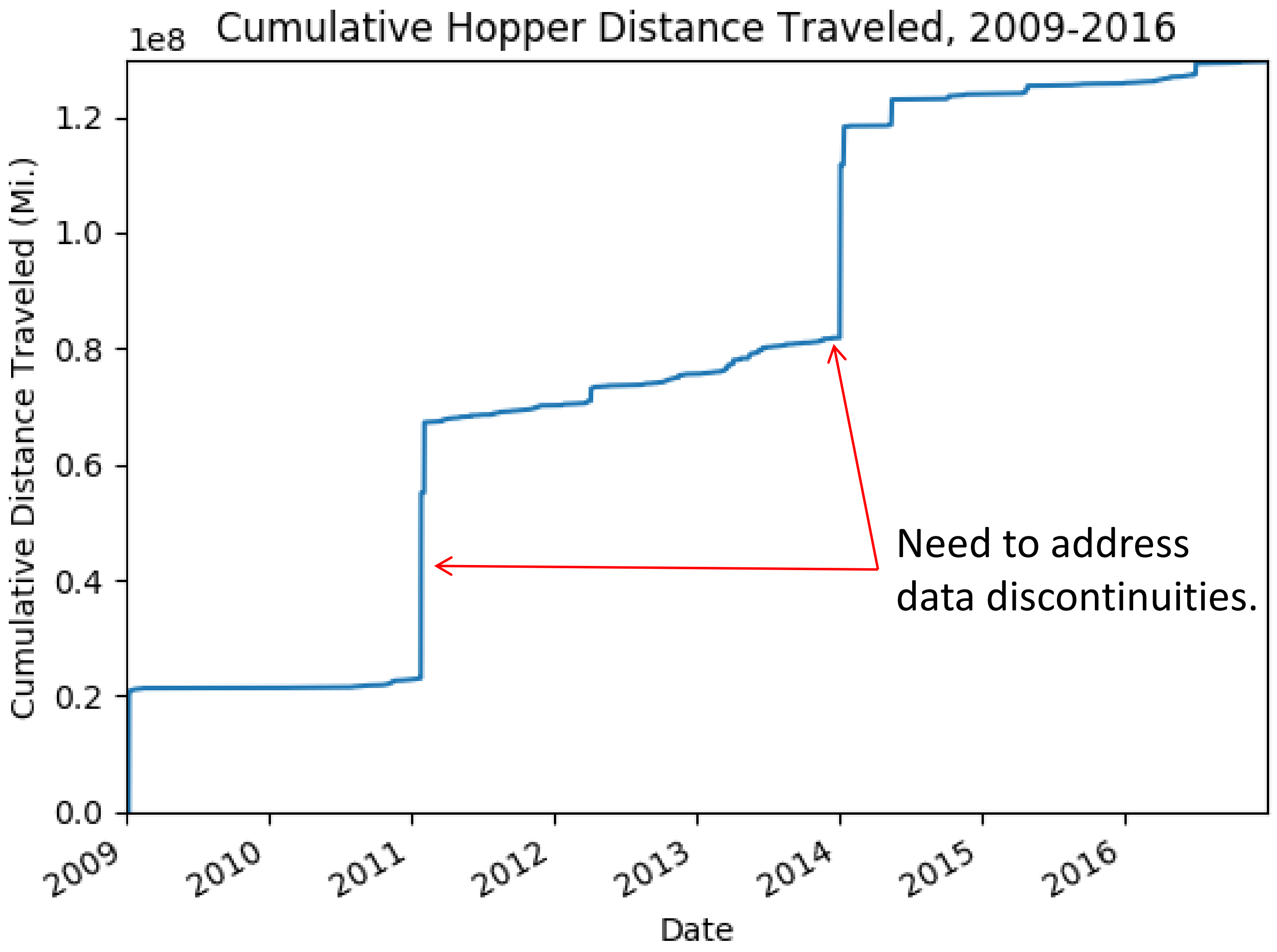
Implicit fixed sequence, unique only for individual plant, eg. PLANT A, B may report at same time, one or many contracts

Invalid when Atomic Expectation violated. Dredge can’t be two places at once.

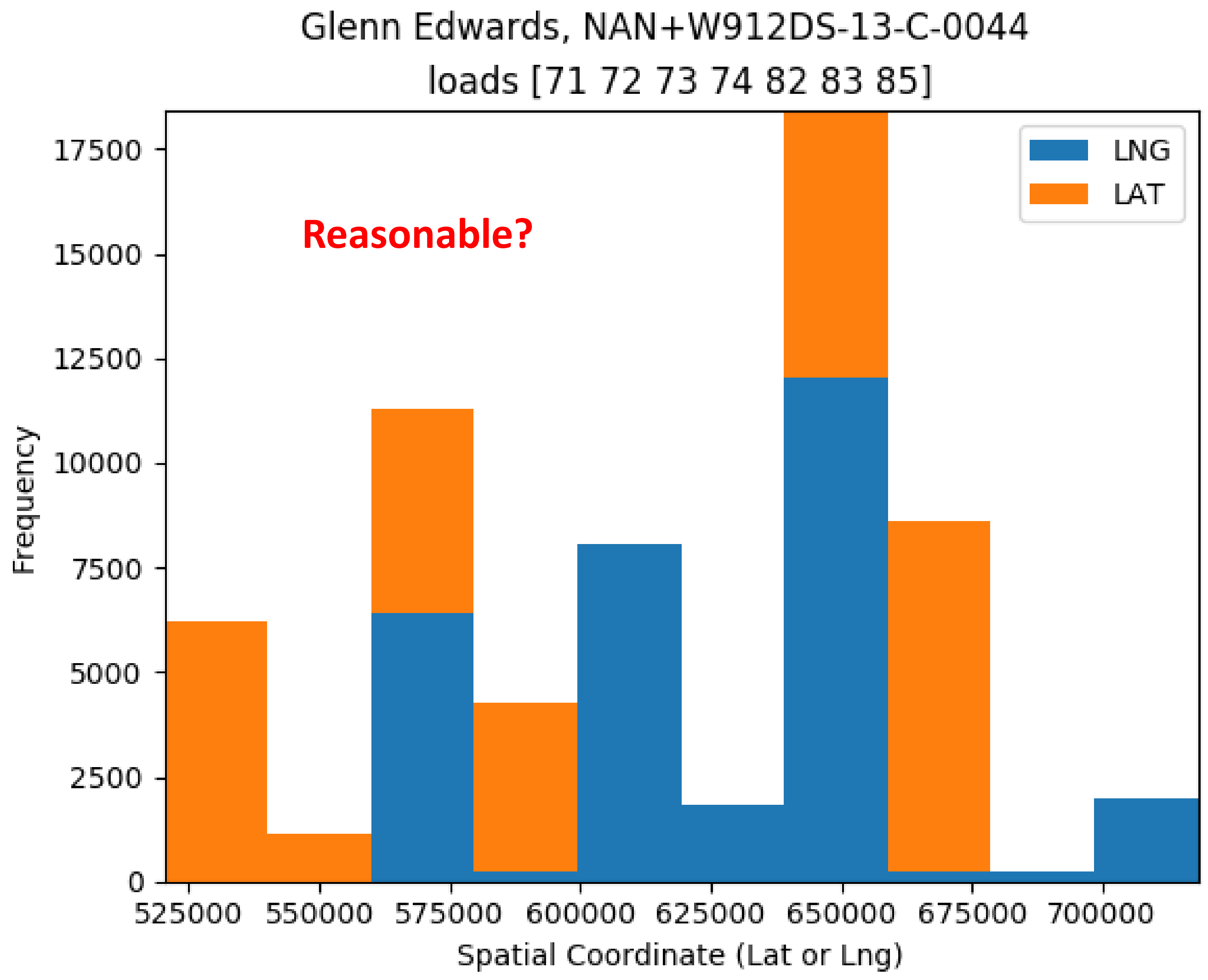
“Sequential”, “unique” only for individual plant, on individual contract, eg. Plant A may report duplicate loads on separate contracts.


DQM Hopper Load Duration, 2009-2016
# of Loads
Load Transit Duration (min)
0
500
1000
1500
2000
2500
3000
3500
0
200
400
600
800
1000
1200
1400
Reasonable?
max duration: 70 days 11:39:38
Expect loads to have duration <1 day, several hours at most.
Expect loads to have non-0 duration.


DQM Hopper Loads, 2009-2016
max distance: 42515095.49114351 mi
Reasonable?
# of Loads
Load Transit Distance (mi)
0
1000
2000
3000
4000
5000
0
10
20
30
40

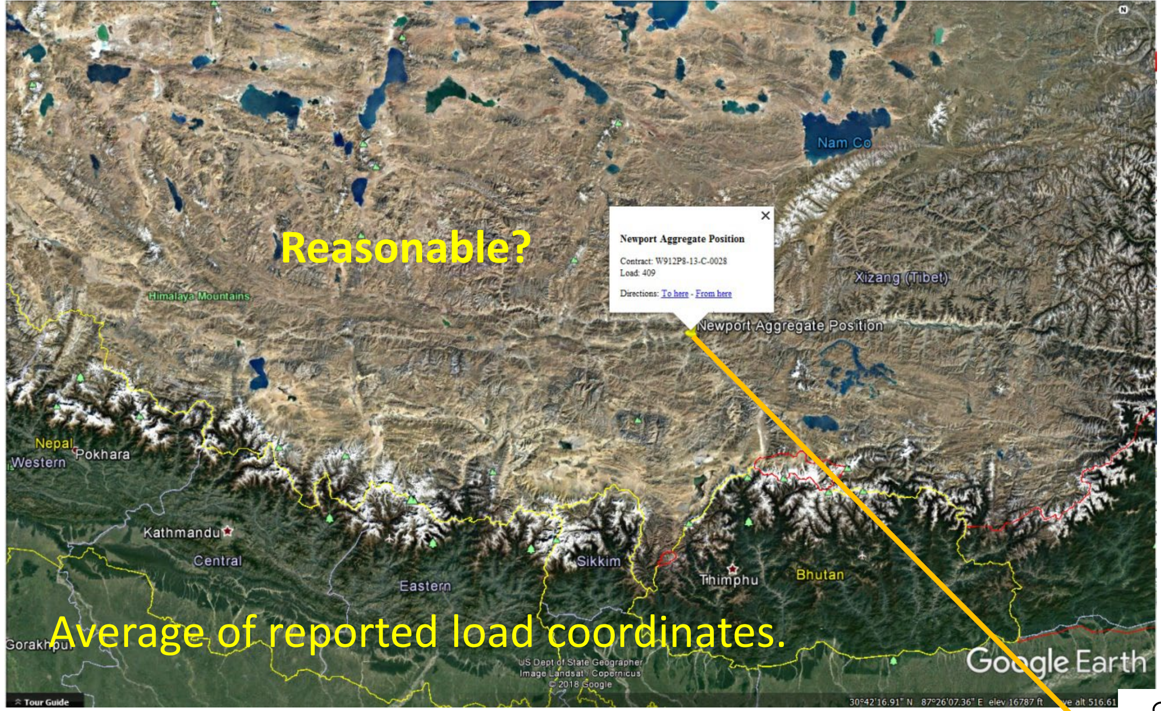


Cumulative Hopper Distance Traveled, 2009-2016
1e8
Cumulative Distance Traveled (Mi.)
0.0
0.2
0.4
0.6
0.8
1.0
1.2
2009
2010
2011
2012
2013
2014
2015
2016
Date
Need to address data discontinuities.



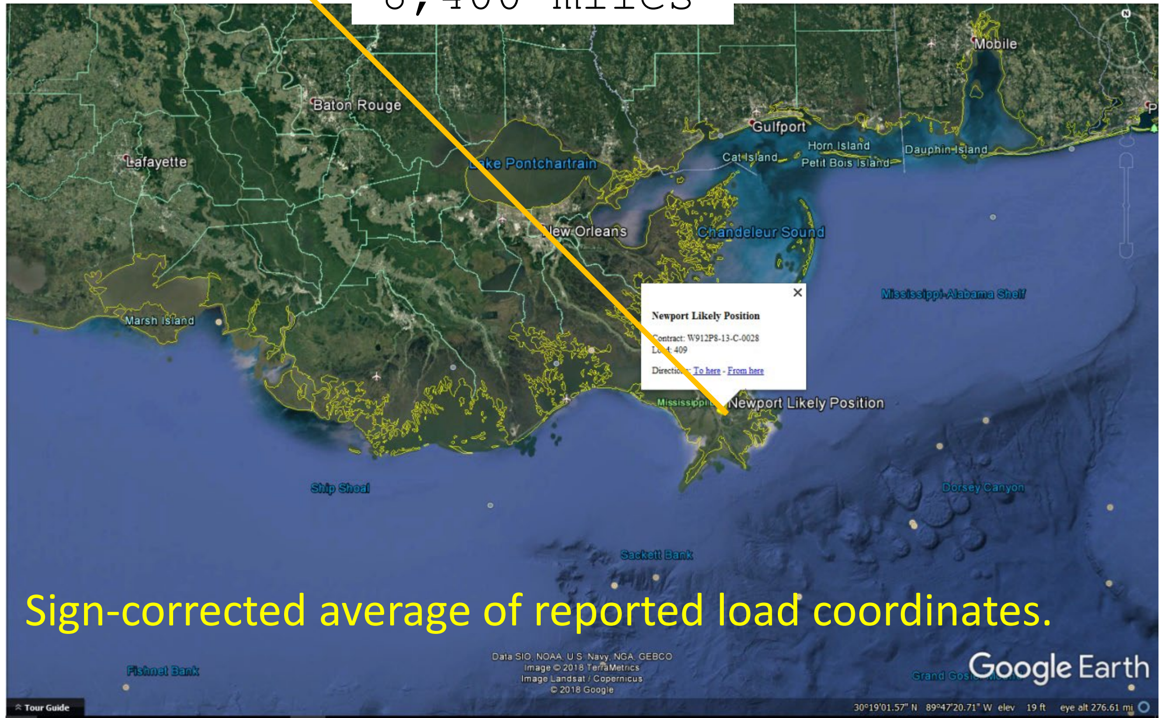
Max LAT/LNG

Spatial coordinates being reported/stored outside of permissible values. Possibly associated with state plane or alternate coordinate systems.


Reasonable?
Newport Aggregate Position
Contract: W912P8-13-C-0028
Load: 409
Directions: To here - From here
Nam Co
Xizang (Tibet)
Himalaya Mountains
Newport Aggregate Position
Nepal
Western
Pokhara
Kathmandu
Central
Eastern
Sikkim
Thimphu
Bhutan
Gorakhpur
Average of reported load coordinates.
Google Earth

Dredge Newport aggregate position,
Contract: W912P8-13-C-0028
Load: 409
LNG 89.259768
LAT 29.172218

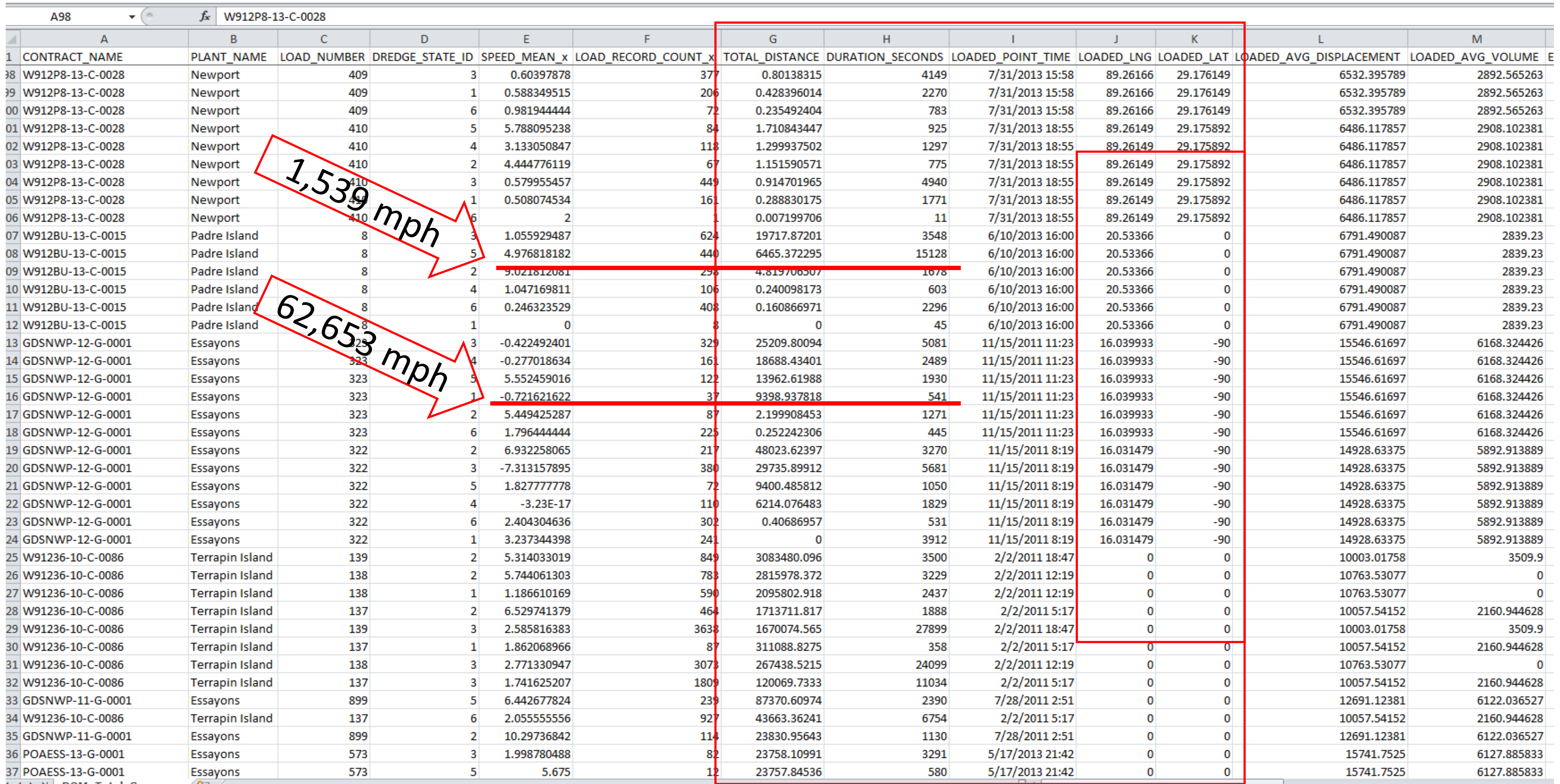
~8,400 miles



Dredge Newport likely position,
Contract: W912P8-13-C-0028
Load: 409
Project: Southwest Pass
LNG -89.259768
LAT 29.172218

General Data Inconsistency

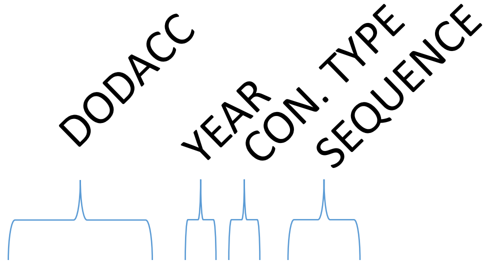
Mean speed can’t be negative.
Computed Speed physically unrealistic.
Computed speed and recorded speed don’t match (good in this case?)



	CONTRACT_NAME	PLANT_NAME	LOAD_NUMBER	DREDGE_STATE_ID	SPEED_MEAN_x	LOAD_RECORD_COUNT_x	TOTAL_DISTANCE	DURATION_SECONDS	LOADED_POINT_TIME	LOADED_LNG	LOADED_LAT	LOADED_AVG_DISPLACEMENT	LOADED_AVG_VOLUME
98	W912P8-13-C-0028	Newport	409	3	0.60397878	377	0.80138315	4149	7/31/2013 15:58	89.26166	29.176149	6532.395789	2892.565263
99	W912P8-13-C-0028	Newport	409	1	0.588349515	206	0.428396014	2270	7/31/2013 15:58	89.26166	29.176149	6532.395789	2892.565263
00	W912P8-13-C-0028	Newport	409	6	0.981944444	72	0.235492404	783	7/31/2013 15:58	89.26166	29.176149	6532.395789	2892.565263
01	W912P8-13-C-0028	Newport	410	5	5.788095238	84	1.710843447	925	7/31/2013 18:55	89.26149	29.175892	6486.117857	2908.102381
02	W912P8-13-C-0028	Newport	410	4	3.133050847	118	1.299937502	1297	7/31/2013 18:55	89.26149	29.175892	6486.117857	2908.102381
03	W912P8-13-C-0028	Newport	410	2	4.444776119	67	1.151590571	775	7/31/2013 18:55	89.26149	29.175892	6486.117857	2908.102381
04	W912P8-13-C-0028	Newport	410	3	0.579955457	449	0.914701965	4940	7/31/2013 18:55	89.26149	29.175892	6486.117857	2908.102381
05	W912P8-13-C-0028	Newport	410	1	0.508074534	161	0.288830175	1771	7/31/2013 18:55	89.26149	29.175892	6486.117857	2908.102381
06	W912P8-13-C-0028	Newport	410	6	2	1	0.007199706	11	7/31/2013 18:55	89.26149	29.175892	6486.117857	2908.102381
07	W912BU-13-C-0015	Padre Island	8	3	1.055929487	624	19717.87201	3548	6/10/2013 16:00	20.53366	0	6791.490087	2839.23
08	W912BU-13-C-0015	Padre Island	8	5	4.976818182	440	6465.372295	15128	6/10/2013 16:00	20.53366	0	6791.490087	2839.23
09	W912BU-13-C-0015	Padre Island	8	2	9.021812081	298	4.819706507	1678	6/10/2013 16:00	20.53366	0	6791.490087	2839.23
10	W912BU-13-C-0015	Padre Island	8	4	1.047169811	106	0.240098173	603	6/10/2013 16:00	20.53366	0	6791.490087	2839.23
11	W912BU-13-C-0015	Padre Island	8	6	0.246323529	408	0.160866971	2296	6/10/2013 16:00	20.53366	0	6791.490087	2839.23
12	W912BU-13-C-0015	Padre Island	8	1	0	8	0	45	6/10/2013 16:00	20.53366	0	6791.490087	2839.23
13	GDSNWP-12-G-0001	Essayons	323	3	-0.422492401	329	25209.80094	5081	11/15/2011 11:23	16.039933	-90	15546.61697	6168.324426
14	GDSNWP-12-G-0001	Essayons	323	4	-0.277018634	161	18688.43401	2489	11/15/2011 11:23	16.039933	-90	15546.61697	6168.324426
15	GDSNWP-12-G-0001	Essayons	323	5	5.552459016	122	13962.61988	1930	11/15/2011 11:23	16.039933	-90	15546.61697	6168.324426
16	GDSNWP-12-G-0001	Essayons	323	1	-0.721621622	37	9398.937818	541	11/15/2011 11:23	16.039933	-90	15546.61697	6168.324426
17	GDSNWP-12-G-0001	Essayons	323	2	5.449425287	87	2.199908453	1271	11/15/2011 11:23	16.039933	-90	15546.61697	6168.324426
18	GDSNWP-12-G-0001	Essayons	323	6	1.796444444	225	0.252242306	445	11/15/2011 11:23	16.039933	-90	15546.61697	6168.324426
19	GDSNWP-12-G-0001	Essayons	322	2	6.932258065	217	48023.62397	3270	11/15/2011 8:19	16.031479	-90	14928.63375	5892.913889
20	GDSNWP-12-G-0001	Essayons	322	3	-7.313157895	380	29735.89912	5681	11/15/2011 8:19	16.031479	-90	14928.63375	5892.913889
21	GDSNWP-12-G-0001	Essayons	322	5	1.827777778	72	9400.485812	1050	11/15/2011 8:19	16.031479	-90	14928.63375	5892.913889
22	GDSNWP-12-G-0001	Essayons	322	4	-3.23E-17	110	6214.076483	1829	11/15/2011 8:19	16.031479	-90	14928.63375	5892.913889
23	GDSNWP-12-G-0001	Essayons	322	6	2.404304636	302	0.40686957	531	11/15/2011 8:19	16.031479	-90	14928.63375	5892.913889
24	GDSNWP-12-G-0001	Essayons	322	1	3.237344398	241	0	3912	11/15/2011 8:19	16.031479	-90	14928.63375	5892.913889
25	W91236-10-C-0086	Terrapin Island	139	2	5.314033019	849	3083480.096	3500	2/2/2011 18:47	0	0	10003.01758	3509.9
26	W91236-10-C-0086	Terrapin Island	138	2	5.744061303	783	2815978.372	3229	2/2/2011 12:19	0	0	10763.53077	0
27	W91236-10-C-0086	Terrapin Island	138	1	1.186610169	590	2095802.918	2437	2/2/2011 12:19	0	0	10763.53077	0
28	W91236-10-C-0086	Terrapin Island	137	2	6.529741379	464	1713711.817	1888	2/2/2011 5:17	0	0	10057.54152	2160.944628
29	W91236-10-C-0086	Terrapin Island	139	3	2.585816383	3638	1670074.565	27899	2/2/2011 18:47	0	0	10003.01758	3509.9
30	W91236-10-C-0086	Terrapin Island	137	1	1.862068966	87	311088.8275	358	2/2/2011 5:17	0	0	10057.54152	2160.944628
31	W91236-10-C-0086	Terrapin Island	138	3	2.771330947	3073	267438.5215	24099	2/2/2011 12:19	0	0	10763.53077	0
32	W91236-10-C-0086	Terrapin Island	137	3	1.741625207	1809	120069.7333	11034	2/2/2011 5:17	0	0	10057.54152	2160.944628
33	GDSNWP-11-G-0001	Essayons	899	5	6.442677824	239	87370.60974	2390	7/28/2011 2:51	0	0	12691.12381	6122.036527
34	W91236-10-C-0086	Terrapin Island	137	6	2.055555556	927	43663.36241	6754	2/2/2011 5:17	0	0	10057.54152	2160.944628
35	GDSNWP-11-G-0001	Essayons	899	2	10.29736842	114	23830.95643	1130	7/28/2011 2:51	0	0	12691.12381	6122.036527
36	POAESS-13-G-0001	Essayons	573	3	1.998780488	82	23758.10991	3291	5/17/2013 21:42	0	0	15741.7525	6127.885833
37	POAESS-13-G-0001	Essayons	573	5	5.675	12	23757.84536	580	5/17/2013 21:42	0	0	15741.7525	6127.885833

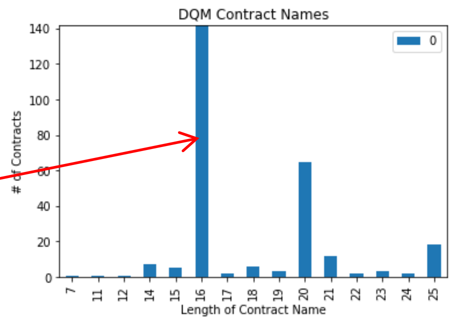
270 Unique DQM Contract #'s

(2009-2016)



Valid: W91236-11-C-0056

(16 Characters)



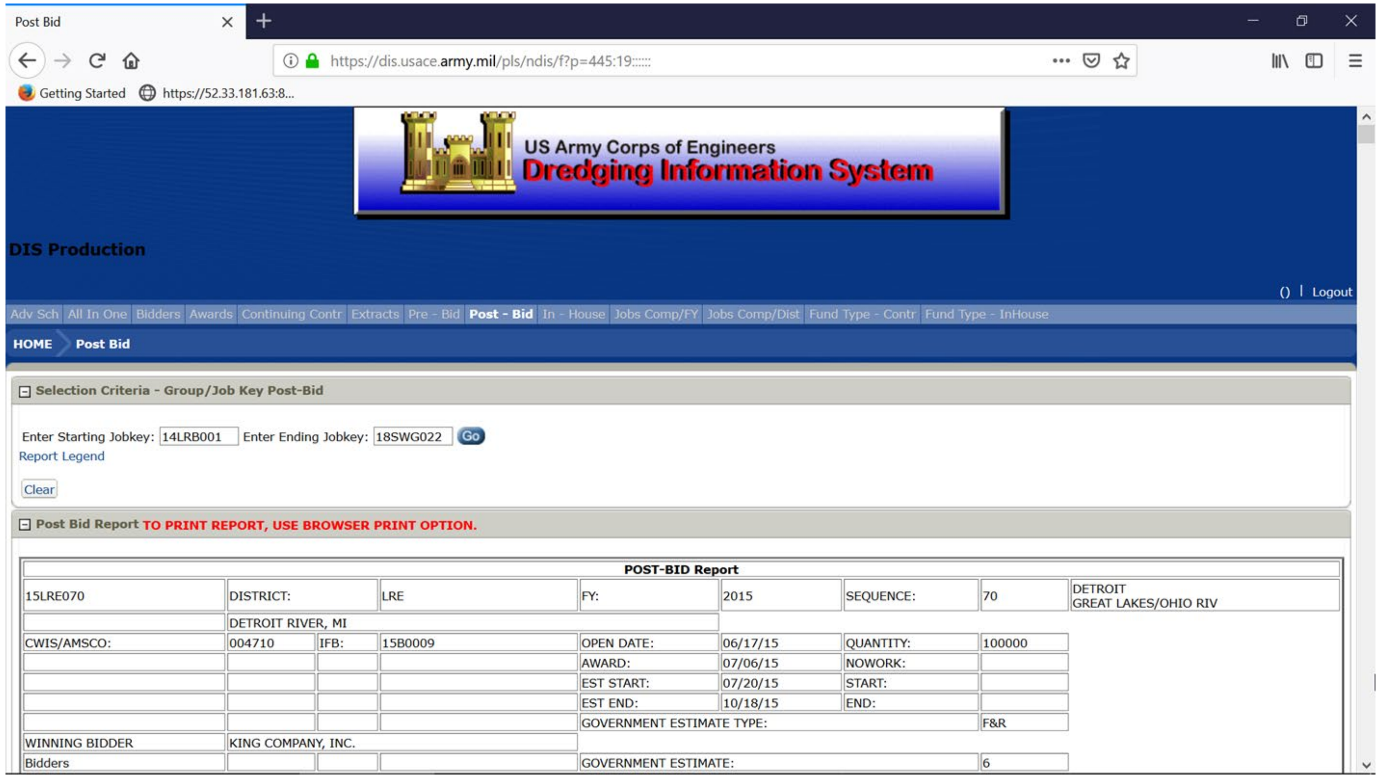
119 Validly Constructed
81 Potentially Valid
69 Invalidly Constructed
1 Annoyingly Invalid

'SAJ+SAJ-1995-3779(SP-LCK)', 'SAJ+W91278-13-D-0007-0007', 'SAJ+W912EP-11-D-0004-0007', 'SAJ+W912EP-11-D-0004-0008', 'SAJ+W912EP-12-D-0003-0003', 'SAJ+W912EP-13-C-0007-0008', 'SAJ+W912EP-13-D-0005-0003', 'SAJ+W912EP-13-D-0005-0004', 'SAJ+W912EP-13-D-0005-0005', 'SAJ+W912EP-13-D-0007-0006', 'SAJ+W912EP-13-D-0007-0007', 'SAJ+W912EP-13-D-0007-0008', 'SAJ+W912EP-13-D-0007-0009', 'SAJ+W912EP-13-D-0007-0010', 'SAJ+W912EP-13-D-0007-0011', 'SAJ+W912EP-13-D-0012-0003', 'SAJ+W912EP-13-D-0012-0004', 'SAJ+W912EP-13-D-0014-0002', 'SAC+W912EP-11-D-0006_SAC', 'SAJ+W912EP-13-D-000-0009', 'SAJ-1992-01720(MOD-LCK)', 'SAJ-2001-05838 (SP-HMM)', 'SAJ-2003-10496 (SP-TSH)', 'MVN+MVN-2011-02539-WPP', 'SAJ-2004-12003(SP-MEP)', 'W912EP-11-D-0004-0002', 'W912EP-11-D-0004-0004', 'W912EP-11-D-0004-0005', 'W912EP-11-D-0004-0009', 'W912EP-11-D-0004_CK01', 'W912EP-12-D-0003-0001', 'W912EP-13-D-0005-0003', 'W912EP-13-D-0005-0005', 'W912EP-13-D-0007-0006', 'W912EP-13-D-0007-0013', 'W912EP-13-D-0008-CV01', 'W912EP-13-D-0012-CK01', 'MVN+GDSMVN-14-G-0001', 'MVN+GDSMVN-15-G-0001', 'MVN+W912P8-14-C-0017', 'MVN+W912P8-14-C-0024', 'MVN+W912P8-15-C-0020', 'MVN+W912P8-15-C-0022', 'MVN+W912P8-15-C-0037', 'MVN+W912P8-16-C-0004', 'NAB+W912DR-13-C-0036', 'NAN+W912DS-13-C-0039', 'NAN+W912DS-13-C-0044', 'NAN+W912DS-13-C-0047', 'NAN+W912DS-13-C-0052', 'NAN+W912DS-14-C-0002', 'NAN+W912DS-14-C-0015', 'NAN+W912DS-14-C-0020', 'NAN+W912DS-15-C-0001', 'NAN+W912DS-15-C-0005', 'NAN+W912DS-15-C-0006', 'NAO+N40085-14-C-8164', 'NAO+W91236-14-C-0014', 'NAO+W91236-14-C-0021', 'NAO+W91236-15-C-0003', 'NAO+W91236-15-C-0033', 'NAP+GDSNAP-14-G-0001', 'NAP+GDSNAP-15-G-0001', 'NAP+GDSNAP-16-G-0001', 'NAP+W912BU-14-C-0008', 'NAP+W912BU-14-C-0013', 'NAP+W912BU-14-C-0015', 'NAP+W912BU-15-C-0003', 'NAP+W912BU-15-C-0007', 'NWP+GDSNWP-14-G-0001', 'NWP+GDSNWP-14-G-0002', 'NWP+GDSNWP-15-G-0001', 'NWP+GDSNWP-15-G-0002', 'NWP+GDSNWP-16-G-0001', 'NWP+GDSNWP-16-G-0002', 'NWP+W9127N-14-C-0018', 'NWP+W9127N-15-C-0006', 'POA+W911KB-12-C-0004', 'POA+W911KB-15-C-0006', 'SAJ+W912EP-11-D-0004', 'SAJ+W912EP-11-D-0006', 'SAJ+W912EP-13-C-0015', 'SAM+W91278-13-D-0024', 'SAM+W91278-14-C-0009', 'SAM+W91278-14-D-0041', 'SAM+W91278-14-D-0087', 'SAS+W912HN-14-C-0001', 'SAS+W912HN-15-C-0003', 'SAS+W912HN-15-C-0005', 'SAS+W912HN-16-C-0001', 'SAW+GDSSAW-15-G-0002', 'SAW+GDSSAW-16-G-0002', 'SAW+W912PM-14-C-0001', 'SAW+W912PM-14-C-0017', 'SWG+SWG-WHEELER-2015', 'SWG+W9126G-13-C-0031', 'SWG+W9126G-13-C-0041', 'SWG+W9126G-14-C-0012', 'SWG+W9126G-14-C-0033', 'SWG+W9126G-15-C-0014', 'SWG+W9126G-15-C-0141', 'W912EP-11-D-0006_SAM', '407-17153-CIMT-1106', 'NWS+NWS-ESS-2014-01', 'NWS+NWS-YAQ-2014-01', 'MVN-2010-01066-ETT', 'MVN-2011-02539-WPP', 'NAN+W912DS-13-0052', 'SAJ+0300119-001-JC', 'SAJ+SAJ-1991-30682', 'SWG+SWG-2004-02311', 'NAO+NAO-2013-1502', 'SAM-2011-0687-DEM', 'GDSMVN-13-G-0001', 'GDSMVN-16-G-0001', 'GDSNAP-12-G-0001', 'GDSNAP-13-G-0001', 'GDSNAP-16-G-0001', 'GDSNWP-10-G-0001', 'GDSNWP-11-G-0001', 'GDSNWP-11-G-0002', 'GDSNWP-12-G-0001', 'GDSNWP-12-G-0002', 'GDSNWP-13-G-0001', 'GDSNWP-13-G-0002', 'GDSNWP-14-G-0001', 'GDSNWP-14-G-0002', 'GDSNWP-16-G-0001', 'GDSNWP-16-G-0002', 'GDSNWP-17-G-0002', 'GDSSAW-12-G-0002', 'GDSSAW-13-G-0002', 'GDSSAW-16-G-0002', 'GDSSAW-17-G-0002', 'POA+ESS-2014-POA', 'POAESS-13-G-0001', 'W911KB-08-C-0002', 'W911KB-12-C-0004', 'W911KB-15-C-0006', 'W91236-10-C-0086', 'W91236-11-C-0027', 'W91236-11-C-0056', 'W91236-12-C-0041', 'W91236-12-C-0042', 'W91236-13-C-0013', 'W9126G-13-C-0031', 'W9126G-13-C-0041', 'W9126G-15-C-0250', 'W9126G-16-C-0031', 'W9126G-16-C-0035', 'W9126G-16-C-0050', 'W9126G-16-C-0051', 'W91278-10-D-0035', 'W91278-10-D-0051', 'W91278-10-D-0099', 'W91278-11-D-0003', 'W91278-12-D-0012', 'W91278-13-D-0001', 'W91278-13-D-0005', 'W91278-13-D-0024', 'W91278-16-D-0041', 'W9127N-11-C-0015', 'W9127N-12-C-0008', 'W9127N-13-C-0008', 'W9127N-16-C-0007', 'W912BU-11-C-0003', 'W912BU-11-C-0005', 'W912BU-11-C-0034', 'W912BU-12-C-0034', 'W912BU-12-C-0046', 'W912BU-13-C-0001', 'W912BU-13-C-0015', 'W912BU-13-C-0025', 'W912BU-14-C-0015', 'W912BU-15-C-0007', 'W912BU-16-C-0033', 'W912BU-16-C-0058', 'W912DR-10-C-0088', 'W912DS-09-C-0023', 'W912DS-10-C-0002', 'W912DS-11-C-0012', 'W912DS-11-C-0024', 'W912DS-11-C-0025', 'W912DS-12-C-0002', 'W912DS-12-C-0021', 'W912DS-12-C-0026', 'W912DS-12-C-0028', 'W912DS-13-C-0033', 'W912DS-13-C-0039', 'W912DS-13-C-0044', 'W912DS-15-C-0001', 'W912DS-15-C-0005', 'W912DS-15-C-0014', 'W912DS-16-C-0019', 'W912EP-10-C-0005', 'W912EP-10-C-0038', 'W912EP-10-C-0040', 'W912EP-11-C-0030', 'W912EP-11-C-0032', 'W912EP-11-D-0004', 'W912EP-11-D-0006', 'W912EP-13-C-0001', 'W912EP-13-C-0015', 'W912EP-13-D-0007', 'W912EP-16-C-0014', 'W912HN-07-C-0053', 'W912HN-09-D-0002', 'W912HN-10-D-0012', 'W912HN-11-C-0008', 'W912HN-11-D-0006', 'W912HN-12-C-0001', 'W912HN-13-C-0001', 'W912HN-14-C-0001', 'W912HN-15-C-0005', 'W912HN-16-C-0001', 'W912HP-15-C-0006', 'W912HY-10-C-0025', 'W912HY-11-C-0007', 'W912HY-11-C-0016', 'W912HY-12-C-0003', 'W912HY-12-C-0008', 'W912HY-12-C-0016', 'W912HY-12-C-0017', 'W912HY-12-C-0023', 'W912P8-10-C-0030', 'W912P8-11-C-0001', 'W912P8-11-C-0031', 'W912P8-11-C-0034', 'W912P8-11-C-0040', 'W912P8-11-C-0045', 'W912P8-12-C-0008', 'W912P8-12-C-0013', 'W912P8-12-C-0017', 'W912P8-12-C-0029', 'W912P8-12-C-0058', 'W912P8-13-C-0014', 'W912P8-13-C-0025', 'W912P8-13-C-0028', 'W912P8-13-C-0029', 'W912P8-13-C-0033', 'W912P8-16-C-0004', 'W912P8-16-C-0017', 'W912P8-16-C-0018', 'W912P8-16-C-0020', 'W912P8-16-C-0024', 'W912P8-16-C-0057', 'W912PL-12-C-0026', 'W912PM-11-B-0005', 'W912PM-12-C-0001', 'W912PM-12-C-0017', 'W912PM-13-C-0001', 'W912PM-16-C-0011', 'W912PN-11-C-0001', 'W912WJ-11-C-0008', 'W912WJ-16-C-0001', 'MVN+MVN-LA-2013', 'NWS-ESS-2012-01', 'NWS-ESS-2013-01', 'NWS-YAQ-2012-01', 'NWS-YAQ-2013-01', '0300119-001-JC', 'SAJ-1992-01740', 'SAJ-1994-03952', 'SAJ-2008-00895', 'SAJ-2009-03448', 'SAW-2006-40282', 'SAW-2012-00026', 'CLW-2011-001', 'MVN-LA-2013', 'UNKNOWN'

Dredging Information System (DIS)

https://dis.usace.army.mil/pls/ndis/f?p=445:19::::::

- DIS retains administrative information regarding dredging contracts:
  - Contract numbers, bidders, successful bidder, bid volume, bid unit price.
  - Plant, volume removed, total cost, start and end date.


Post Bid
https://dis.usace.army.mil/pls/ndis/f?p=445:19:::::
Getting Started
https://52.33.181.63:8...
US Army Corps of Engineers
Dredging Information System
DIS Production
() | Logout
Adv Sch | All In One | Bidders | Awards | Continuing Contr | Extracts | Pre - Bid | Post - Bid | In - House | Jobs Comp/FY | Jobs Comp/Dist | Fund Type - Contr | Fund Type - InHouse
HOME | Post Bid

Selection Criteria - Group/Job Key Post-Bid

Enter Starting Jobkey: 14LRB001 Enter Ending Jobkey: 18SWG022 Go

Report Legend

Clear

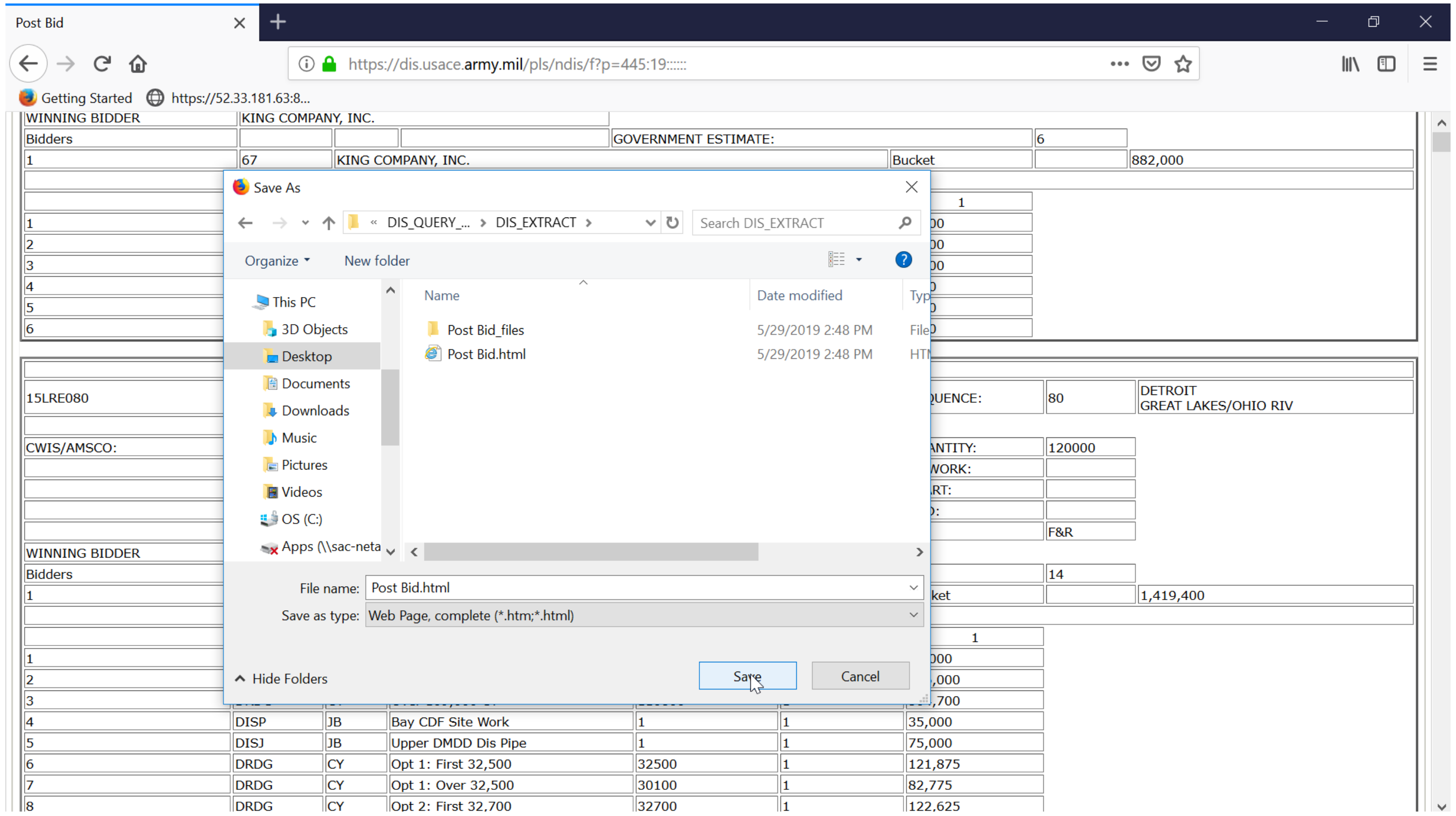
Post Bid Report TO PRINT REPORT, USE BROWSER PRINT OPTION.

POST-BID Report

15LRE070	DISTRICT:		LRE	FY:	2015	SEQUENCE:	70	DETROIT GREAT LAKES/OHIO RIV
	DETROIT RIVER, MI							
CWIS/AMSCO:	004710	IFB:	15B0009	OPEN DATE:	06/17/15	QUANTITY:	100000	
				AWARD:	07/06/15	NOWORK:		
				EST START:	07/20/15	START:		
				EST END:	10/18/15	END:		
				GOVERNMENT ESTIMATE TYPE:			F&R	
WINNING BIDDER	KING COMPANY, INC.							
Bidders				GOVERNMENT ESTIMATE:			6	

DIS output had to be manually interpreted.

Data stored with positional reference, not Class/Id! Everything named “table”.



Post Bid

https://dis.usace.army.mil/pls/ndis/f?p=445:19::::::

Getting Started https://52.33.181.63:8...

WINNING BIDDER	KING COMPANY, INC.					
Bidders				GOVERNMENT ESTIMATE:	6	
1	67	KING COMPANY, INC.		Bucket		882,000

15LRE080 ... QUENCE: 80 DETROIT GREAT LAKES/OHIO RIV

CWIS/AMSCO: ... ANTITY: 120000 ... WORK: ... RT: ...): ... F&R

WINNING BIDDER

Bidders ... 14

1 ... ket ... 1,419,400

4	DISP	JB	Bay CDF Site Work	1	1	35,000
5	DISJ	JB	Upper DMDD Dis Pipe	1	1	75,000
6	DRDG	CY	Opt 1: First 32,500	32500	1	121,875
7	DRDG	CY	Opt 1: Over 32,500	30100	1	82,775
8	DRDG	CY	Opt 2: First 32,700	32700	1	122,625

Save As

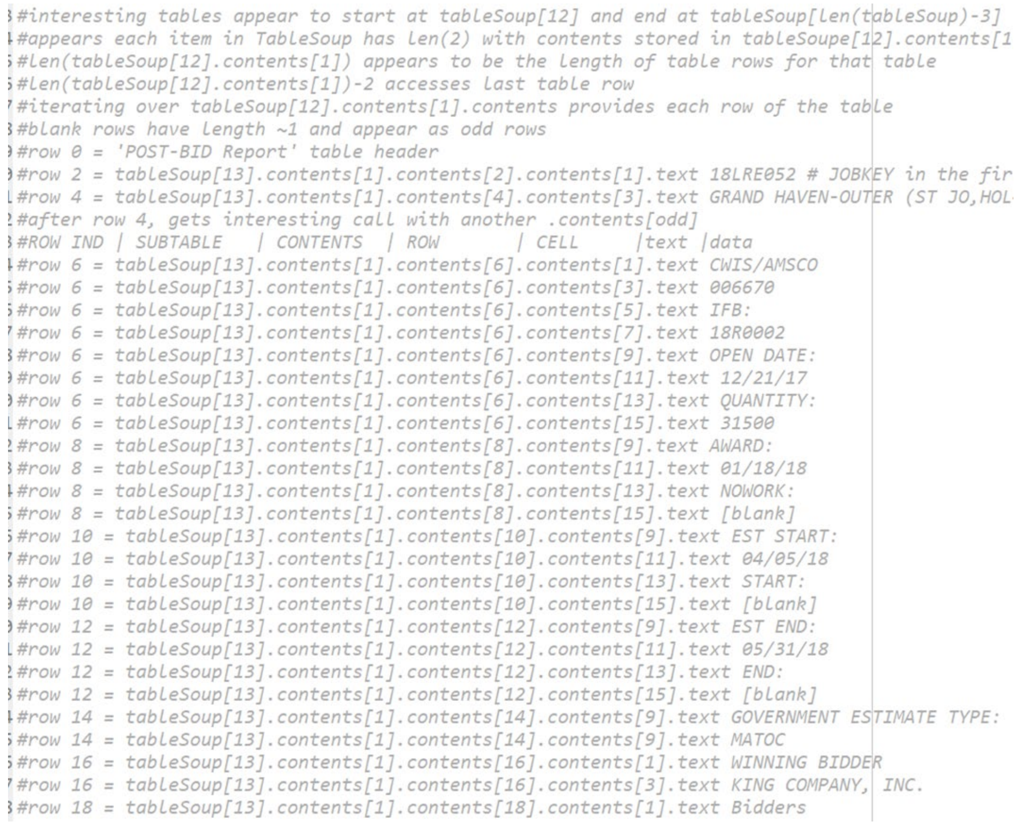
« DIS_QUERY_... › DIS_EXTRACT ›

Search DIS_EXTRACT

Name	Date modified	Typ
Post Bid_files	5/29/2019 2:48 PM	File
Post Bid.html	5/29/2019 2:48 PM	HTI

File name: Post Bid.html

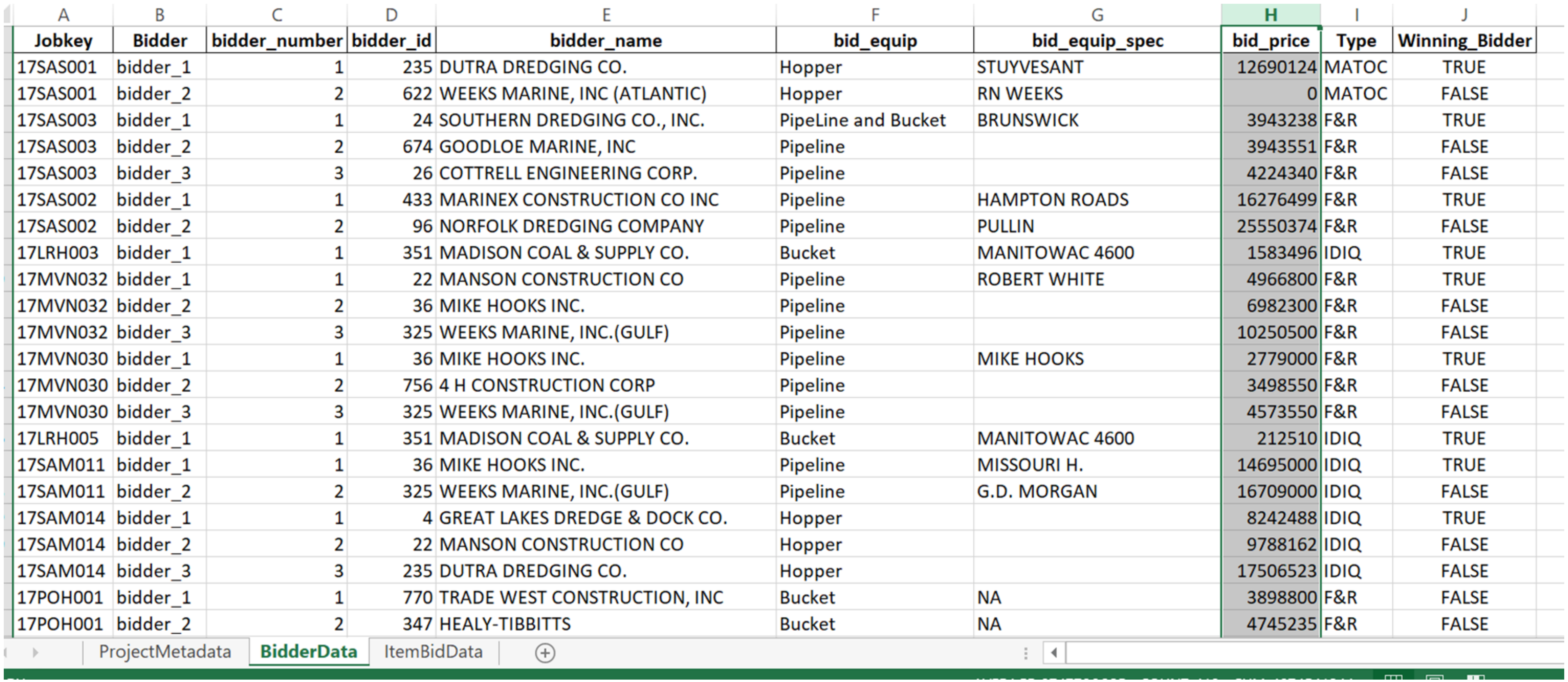
Save as type: Web Page, complete (*.htm;*.html)



```
#interesting tables appear to start at tableSoup[12] and end at tableSoup[len(tableSoup)-3]
#appears each item in TableSoup has len(2) with contents stored in tableSoupe[12].contents[1
#len(tableSoup[12].contents[1]) appears to be the length of table rows for that table
#len(tableSoup[12].contents[1])-2 accesses last table row
#iterating over tableSoup[12].contents[1].contents provides each row of the table
#blank rows have length ~1 and appear as odd rows
#row 0 = 'POST-BID Report' table header
#row 2 = tableSoup[13].contents[1].contents[2].contents[1].text 18LRE052 # JOBKEY in the fir
#row 4 = tableSoup[13].contents[1].contents[4].contents[3].text GRAND HAVEN-OUTER (ST JO,HOL
#after row 4, gets interesting call with another .contents[odd]
#ROW IND | SUBTABLE    | CONTENTS  | ROW        | CELL      |text |data
#row 6 = tableSoup[13].contents[1].contents[6].contents[1].text CWIS/AMSCO
#row 6 = tableSoup[13].contents[1].contents[6].contents[3].text 006670
#row 6 = tableSoup[13].contents[1].contents[6].contents[5].text IFB:
#row 6 = tableSoup[13].contents[1].contents[6].contents[7].text 18R0002
#row 6 = tableSoup[13].contents[1].contents[6].contents[9].text OPEN DATE:
#row 6 = tableSoup[13].contents[1].contents[6].contents[11].text 12/21/17
#row 6 = tableSoup[13].contents[1].contents[6].contents[13].text QUANTITY:
#row 6 = tableSoup[13].contents[1].contents[6].contents[15].text 31500
#row 8 = tableSoup[13].contents[1].contents[8].contents[9].text AWARD:
#row 8 = tableSoup[13].contents[1].contents[8].contents[11].text 01/18/18
#row 8 = tableSoup[13].contents[1].contents[8].contents[13].text NOWORK:
#row 8 = tableSoup[13].contents[1].contents[8].contents[15].text [blank]
#row 10 = tableSoup[13].contents[1].contents[10].contents[9].text EST START:
#row 10 = tableSoup[13].contents[1].contents[10].contents[11].text 04/05/18
#row 10 = tableSoup[13].contents[1].contents[10].contents[13].text START:
#row 10 = tableSoup[13].contents[1].contents[10].contents[15].text [blank]
#row 12 = tableSoup[13].contents[1].contents[12].contents[9].text EST END:
#row 12 = tableSoup[13].contents[1].contents[12].contents[11].text 05/31/18
#row 12 = tableSoup[13].contents[1].contents[12].contents[13].text END:
#row 12 = tableSoup[13].contents[1].contents[12].contents[15].text [blank]
#row 14 = tableSoup[13].contents[1].contents[14].contents[9].text GOVERNMENT ESTIMATE TYPE:
#row 14 = tableSoup[13].contents[1].contents[14].contents[9].text MATOC
#row 16 = tableSoup[13].contents[1].contents[16].contents[1].text WINNING BIDDER
#row 16 = tableSoup[13].contents[1].contents[16].contents[3].text KING COMPANY, INC.
#row 18 = tableSoup[13].contents[1].contents[18].contents[1].text Bidders
```

Target storage was Excel.

Not elegant, but culturally acceptable.



Jobkey	Bidder	bidder_number	bidder_id	bidder_name	bid_equip	bid_equip_spec	bid_price	Type	Winning_Bidder
17SAS001	bidder_1	1	235	DUTRA DREDGING CO.	Hopper	STUYVESANT	12690124	MATOC	TRUE
17SAS001	bidder_2	2	622	WEEKS MARINE, INC (ATLANTIC)	Hopper	RN WEEKS	0	MATOC	FALSE
17SAS003	bidder_1	1	24	SOUTHERN DREDGING CO., INC.	PipeLine and Bucket	BRUNSWICK	3943238	F&R	TRUE
17SAS003	bidder_2	2	674	GOODLOE MARINE, INC	Pipeline		3943551	F&R	FALSE
17SAS003	bidder_3	3	26	COTTRELL ENGINEERING CORP.	Pipeline		4224340	F&R	FALSE
17SAS002	bidder_1	1	433	MARINEX CONSTRUCTION CO INC	Pipeline	HAMPTON ROADS	16276499	F&R	TRUE
17SAS002	bidder_2	2	96	NORFOLK DREDGING COMPANY	Pipeline	PULLIN	25550374	F&R	FALSE
17LRH003	bidder_1	1	351	MADISON COAL & SUPPLY CO.	Bucket	MANITOWAC 4600	1583496	IDIQ	TRUE
17MVN032	bidder_1	1	22	MANSON CONSTRUCTION CO	Pipeline	ROBERT WHITE	4966800	F&R	TRUE
17MVN032	bidder_2	2	36	MIKE HOOKS INC.	Pipeline		6982300	F&R	FALSE
17MVN032	bidder_3	3	325	WEEKS MARINE, INC.(GULF)	Pipeline		10250500	F&R	FALSE
17MVN030	bidder_1	1	36	MIKE HOOKS INC.	Pipeline	MIKE HOOKS	2779000	F&R	TRUE
17MVN030	bidder_2	2	756	4 H CONSTRUCTION CORP	Pipeline		3498550	F&R	FALSE
17MVN030	bidder_3	3	325	WEEKS MARINE, INC.(GULF)	Pipeline		4573550	F&R	FALSE
17LRH005	bidder_1	1	351	MADISON COAL & SUPPLY CO.	Bucket	MANITOWAC 4600	212510	IDIQ	TRUE
17SAM011	bidder_1	1	36	MIKE HOOKS INC.	Pipeline	MISSOURI H.	14695000	IDIQ	TRUE
17SAM011	bidder_2	2	325	WEEKS MARINE, INC.(GULF)	Pipeline	G.D. MORGAN	16709000	IDIQ	FALSE
17SAM014	bidder_1	1	4	GREAT LAKES DREDGE & DOCK CO.	Hopper		8242488	IDIQ	TRUE
17SAM014	bidder_2	2	22	MANSON CONSTRUCTION CO	Hopper		9788162	IDIQ	FALSE
17SAM014	bidder_3	3	235	DUTRA DREDGING CO.	Hopper		17506523	IDIQ	FALSE
17POH001	bidder_1	1	770	TRADE WEST CONSTRUCTION, INC	Bucket	NA	3898800	F&R	FALSE
17POH001	bidder_2	2	347	HEALY-TIBBITTS	Bucket	NA	4745235	F&R	FALSE

DIS jobkey to contract #?

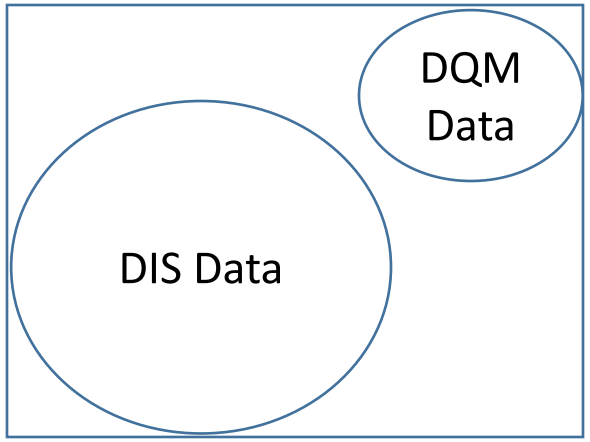
DIS Data

JOBKEY	CONTRACT	IFB
04MVN412	04C0030	03B0082
04SWG022		04B0019
04NWS004	03C0020	03B0013
03NWS004	03C0016	03b0012
03POA005	03C0011	03-R-08
03POA006	03C0013	03R0011
04NAP012	0020	0020
03NAP005		0021
04MVN411	04C0027	03B0081
04NWS001		04B0004
04NWS002	04C0016	04B0008
04NWS003		
03NWS002	03C0021	03B0015
03NWS003	03C0001	02b0013
91SWG007	91C0027	7
91SWG013	91C0024	14
11SWG008		
11SWG011		
11SWG014	11C0025	11B0012
11LRB006		
11NAO007	11C0030	B0006
11NAO012	11C0056	11B0016
12SAC002		11B0001
10NAN011	10C0024	10B0012
11MVN135	10C0127	07B0042

DQM CONTRACT_NAME Data

'W912EP-11-D-0004-0004‘
'W912EP-11-D-0004-0002’
'W912EP-11-D-0004‘
'W912EP-11-C-0032‘
'W912EP-11-C-0030‘
'UNKNOWN‘
'SWG+W9126G-15-C-0141‘
'SWG+W9126G-15-C-0014‘
'SWG+W9126G-14-C-0033‘
'SWG+W9126G-14-C-0012‘
'SAJ-2004-12003(SP-MEP)‘
'SAJ-2003-10496 (SP-TSH)‘
'SAJ-2001-05838 (SP-HMM)‘
'SAJ-1994-03952‘
'SAJ-1992-01740‘
'SAJ-1992-01720(MOD-LCK)‘
'SAJ+W912EP-13-D-0014-0002‘
'SAJ+W912EP-13-D-0012-0004‘
'SAJ+W912EP-13-D-0012-0003‘
'SAC+W912EP-11-D-0006_SAC‘
POAESS-13-G-0001‘
'POA+W911KB-15-C-0006‘
'POA+W911KB-12-C-0004‘
'POA+ESS-2014-POA‘
'NWS-YAQ-2013-01‘
'NAB+W912DR-13-C-0036‘
'MVN-LA-2013‘
'MVN-2011-02539-WPP‘
'MVN-2010-01066-ETT‘
'MVN+W912P8-16-C-0004‘
'GDSSAW-17-G-0002‘
'GDSSAW-16-G-0002'

These don’t overlap!



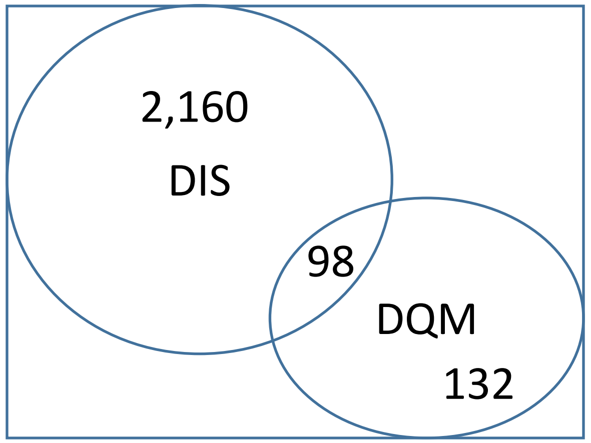
How to translate from one to the other?

DIS jobkey to contract #?

DIS Valid Constructed Contracts: 2,258
DQM Total Contracts: 230
DQM Valid Contracts: 154 (67% of DQM Total)
Overlapping: 98 (43% of DQM Total)

```
'W9123610C0086', 'W9123611C0056', 'W9123612C0041', 'W9123612C0042',
'W9123613C0013', 'W9123614C0021', 'W9127810D0035', 'W9127810D0051',
'W9127814C0009', 'W9127814D0041', 'W9127814D0087', 'W9127N13C0008',
'W9127N14C0018', 'W9127N15C0006', 'W9127N16C0007', 'W912BU11C0003',
'W912BU11C0005', 'W912BU14C0013', 'W912BU16C0033', 'W912BU16C0058',
'W912DR10C0088', 'W912DS09C0023', 'W912DS11C0012', 'W912DS11C0024',
'W912DS11C0025', 'W912DS12C0002', 'W912DS12C0021', 'W912DS12C0026',
'W912DS12C0028', 'W912DS13C0033', 'W912DS13C0039', 'W912DS13C0044',
'W912DS13C0047', 'W912DS13C0052', 'W912DS14C0002', 'W912DS15C0001',
'W912DS15C0005', 'W912DS16C0019', 'W912EP10C0005', 'W912EP10C0038',
'W912EP10C0040', 'W912EP11C0030', 'W912EP11C0032', 'W912EP11D0004',
'W912EP13C0001', 'W912EP13C0015', 'W912HN09D0002', 'W912HN10D0012',
'W912HN11D0006', 'W912HN12C0001', 'W912HN13C0001', 'W912HN14C0001',
'W912HN15C0003', 'W912HN15C0005', 'W912HN16C0001', 'W912HP15C0006',
'W912HY10C0025', 'W912HY11C0007', 'W912HY11C0016', 'W912HY12C0003',
'W912HY12C0008', 'W912HY12C0016', 'W912HY12C0017', 'W912HY12C0023',
'W912P810C0030', 'W912P811C0001', 'W912P811C0031', 'W912P811C0034',
'W912P811C0040', 'W912P811C0045', 'W912P812C0008', 'W912P812C0013',
'W912P812C0017', 'W912P812C0029', 'W912P812C0058', 'W912P813C0014',
'W912P813C0025', 'W912P813C0028', 'W912P813C0029', 'W912P813C0033',
'W912P814C0017', 'W912P814C0024', 'W912P815C0020', 'W912P815C0022',
'W912P815C0037', 'W912P816C0004', 'W912P816C0017', 'W912P816C0018',
'W912P816C0020', 'W912P816C0024', 'W912P816C0057', 'W912PL12C0026',
'W912PM12C0001', 'W912PM12C0017', 'W912PM13C0001', 'W912PM14C0001',
'W912PM14C0017', 'W912PM16C0011'
```

Good, not great!



But at least we can now look at some DQM data in context with DIS, starting with the total cost expended by activity appearing in both DQM and DIS: $1,047,341,787

Data Challenges to Date

- Bad GPS data
- District/Division info not stored -> impute from contract number
  - Except USACE work doesn’t get contractID.
  - Except MATOC might be used for non-owner districts
  - Contract number doesn’t match other legacy systems (e.g. DIS)
- Project name is available (provides logical grouping)
  - no project name enforcement i.e. ambiguous
- Load/Cycle logic is error prone
- Dredge state is error prone
  - Interstitial dredge states
  - Unknown (dredge is doing something outside the dredging cycle)
  - NULL (not enough sensor input to classify dredge state)