Draw Data

```
Step(
    data_layer : DataLayer,
    current_step : FileID ,
    next_step : Str,
    t_metadata : Ibis.Table,
)->FileID:
    """

    """
    m=data_layer .get_metadata(current_step)
    p=data_layer.get_parameters(
        get_most_recent (
            t_metadata =t_metadata      ,
            thing_type ="Draw Data"     ,
            project    =m["Project"]    ,
        )
    )
    Return data_layer.log(**{
        parameters  = p,
        metadata    = {
                    "Has_Parameters":True,
                    "Has_Metadata":True,
                    "Has_Dataset"   :True,
                    "Has_Metrics"   :True,
                    "Thing Type"    :"Draw Data",
                    "Jurisdiction"  :p["JURISDICTION"],
                    "Project"       :p["Project"],
                    },

        dataset     = "Notional (empty) raw data"
        metrics     = {
                     "Dataset_Size": n,      How many records are in the dataset
                     "Schema_Length": m,     How many features are in the dataset
                     "Creation_Time": t,     How long it took to get this dataset
                    },
    })
```

Model Ready

```
Step(                                           feature_map={
    data_layer: DataLayer,                        K:V
    current_step: FileID ,                        for K,v in
    next_step: Str,                               zip(
    t_metadata: Ibis.Table,                          ["feat_{i}" for i in range(100)],
)->FileID:                                           [rng.normal(.6,.3,100).clip(1,0)],
    """                                           )
                                                }
    """
    m= data_layer .get_metadata(current_step)
    dataset_fileid =data_layer.get_metadata(
        get_most_recent(

            t_metadata =t_metadata     ,
            thing_type ="DrawData"     ,
            project    =m["Project"]   ,
        )["FileID"]
    )

    dataset_features = get_random_features(
        feature_map=feature_map,
        rng= rng,
    )


    Return data_layer.log(**{
        parameters = {"DATASET_FILEID": dataset_fileid,
                      "DATASET_FEATURES": dataset_features,
                     },
        metadata   = {
                       "Has_Parameters": True,
                       "Has_Metadata"  : True,
                       "Has_Dataset"   : True,
                       "Has_Metrics"   : True,
                       "Thing Type"    : "ModelReady",
                       "Jurisdiction"  : m["JURISDICTION"],
                       "Project"       : m["Project"],
                     },

        dataset    = "Notional empty model ready data.",
        metrics    = {
                        "Dataset_Size": n ,      How Many records are in the dataset
                        "Schema_Length": m,      How Many features are in the dataset
                        "Creation_Time": t,      How long it took to get this dataset
                     },
    })
```

Model

```
Step(
    data_layer: DataLayer,
    current_step: FileID,
    next_step: Str,
    t_metadata: Ibis.Table,
)->FileID:
    """

    """
    m= data_layer .get_metadata(current_step)
    dataset_fileid =data_layer.get_metadata(
        get_most_recent(
            t_metadata = t_metadata      ,
            thing_type  ="ModelReady"    ,
            project     = m["Project"]   ,
        )["FileID"]
    )

    Return data_layer.log(**{
        parameters  = {
                "Modeling_Data_FileID" : dataset_fileid ,
                "Tree_Depth"           : U[range(4,9)],
                "Tree_Nodes"           : U[range(15 25)],
                "Split_criterion"      : U["Gini","Entropy"],
                },
        metadata    = {
                "Has_Parameters":True,
                "Has_Metadata":True,
                "Has_Dataset"  :True,
                "Has_Metrics"  :True,
                "Thing Type"   :"Model",
                "Jurisdiction" :m["JURISDICTION"],
                "Project"      :M["Project"],
                },

        dataset     = "Notional (empty) Labels for input.",
        metrics     = {
                "Accuracy":f(features)
                "Precision" :f(features)
                "Recall"    :f(features)
                },
        artifacts   = "filepath_with_model_functional_artifact",
    })
```

Aggregate

```
Step(
    data_layer: DataLayer,
    current_step: FileID ,
    next_step: Str,
    t_metadata: Ibis.Table,
)→FileID:
    """

    """
    m=data_layer .get_metadata(current_step)
    modeling_fileid =data_layer.get_metadata(
        get_most_recent(
            t_metadata = t_metadata        ,
            thing_type  = "Model"          ,
            project     = m["Project"]     ,
        )["FileID"]
    )
    dataset_fileid =data_layer.get_parameters(
        modeling_fileid
    )["Modeling_Data_FileID"]

    Return data_layer.log(**{
        parameters = {
                "Modeling_Output_FileID": dataset_fileid ,
                "Modeling_Data_FileID"  : modeling_fileid ,
                "Agg-Method"            : U["weighted","unweighted"],
               },

        metadata  = {
                "Has_Parameters":True,
                "Has_Metadata" :True,
                "Has_Dataset"  :True,
                "Has_Metrics"  :True,
                "Thing Type"   :"Aggregate",
                "Jurisdiction" :m["JURISDICTION"],
                "Project"      :m["Project"],
               },

        dataset   = "Notional Aggregated Model Output",
        metrics   = {
                "Compression_Ratio": Abs(N(0,3)) ,
               },
        artifacts = "filepath_with_model_functional_artifact",
    })
```

Review

```
Step(
    data_layer: DataLayer,
    current_step: FileID ,
    next_step: Str,
    t_metadata: Ibis.Table,
)->FileID:
    """

    """
    m=data_layer .get_metadata(current_step)
    aggregate_fileid =data_layer.get_metadata(
        get_most_recent (
            t_metadata =t_metadata        ,
            thing_type  ="Aggregate"      ,
            project     =m["Project"]     ,
        )["FileID"]
    )
    Modeling_fileid=data_layer.get_parameters(
        aggregate_fileid
    )["Modeling_Output_FileID"]
    Satisfaction=Bernoulli(
        mean(*data_layer.get_metrics(Modeling_fileid))
    )
    Return data_layer.log(**{
        parameters  ={
                "Modeling_Output_FileID":Modeling_fileid ,
                "Aggregate_Output_FileID":aggregate_fileid ,
                "User"                   :"Steve",
                "User_Comment"           :if(satisfaction) "Yay" else "Ew",
            },
        metadata    ={
                "Has_Parameters":True,
                "Has_Metadata"  :True,
                "Has_Dataset"   :True ,
                "Has_Metrics"   :True,
                "Thing Type"    :"Review",
                "Jurisdiction"  :m["JURISDICTION"],
                "Project"       :m["Project"],
            },


        metrics     ={
                "Satisfaction":Satisfaction,
            },
        artifacts   = "Filepath_with_explicit_review_artifacts",
    })
```

Publish

```
Step(
    data_layer: DataLayer,
    current_step: FileID,
    next_step: Str,
    t_metadata: Ibis.Table,
) -> FileID:
    """

    """
    m= data_layer.get_metadata(current_step)
    review_fileid = data_layer.get_metadata(
        get_most_recent (
            t_metadata = t_metadata      ,
            thing_type  = "Review "      ,
            project     = m["Project"]   ,
        )["FileID"]
    )
    p = data_layer.get_parameters(
        review_fileid
    )

    Return data_layer.log(**{
        parameters = {
                "Modeling_Output_FileID" : p["Modeling_Output_FileID"],
                "Aggregate_Output_FileID" : p["Aggregate_Output_FileID"],
                "Review_Output_FileID"     : p["Review_Output_FileID"],

              },
        metadata   = {
                "Has_Parameters" : True,
                "Has_Metadata"   : True,
                "Has_Dataset"    : True,
                "Has_Metrics"    : True,
                "Thing Type"     : "Publish",
                "Jurisdiction"   : m["JURISIICTION"],
                "Project"        : m["Project"],
              },


        metrics    = {
                "Meets_Review" : Satisfaction,
              },
        artifacts  = "filepath_with_explicit_functional_packaging_Artifacts",
    })
```

```
Step( ):


    Return {
        "parameters" : None
        "metadata"   : None
        "dataset"    : None
        "metrics"    : None
        "embedding"  : None
        "artifacts"  : None
        "function"   : None
    }
```

DrawData
ModelReady
Model
Aggregate
Review
Publish

```
get_most_recent(
    t_metadata: ibis.Table,
    thing_type: str,
    project: str,
) → FileID:
    """
    """
    return t_metadata.filter(
        _.Project == project,
        _.ThingType == thing_type,
        _.FileID == _.FileID.argmax(_.DatasetDate),
    ).select("FileID")

get_random_features(
    feature_map: Mapping[str, float],
    rng: Optional[Generator] = None,
) → List[str]:
    """
    """
    return rng.choice(
        feature_map.Keys(),
        10
    )
```