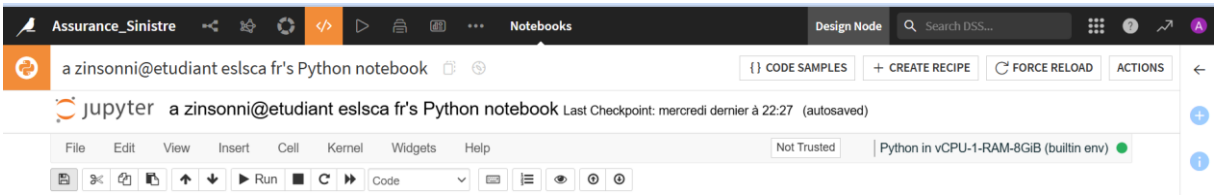
Projet d'analyse des SINISTRES

Outils : Dataiku + Python

I. EDA sous Python (Dataiku)

Exploration du dataset (EDA): **Python**



Importer le dataset chargé sur Dataiku dans l'IDE Dataiku pour python(notebook)

```
In [50]: import matplotlib.pyplot as plt

In [51]: import dataiku
         from dataiku import pandasutils as pdu
         import pandas as pd

In [52]: # Example: Load a DSS dataset as a Pandas dataframe
         mydataset = dataiku.Dataset("Donnees_Assurance_insurance_data")
         df = mydataset.get_dataframe()

         df.head(10)
```

Out[52]:

	TXN_DATE_TIME	TRANSACTION_ID	CUSTOMER_ID	POLICY_NUMBER	POLICY_EFF_DT	LOSS_DT	REPORT_DT	INSURANCE_TYPE	PREMIUM_AMOUNT	CL
0	6/1/20 0:00	TXN00000001	A00003822	PLC00008468	6/23/15	5/16/20	5/21/20	Health	157.13	
1	6/1/20 0:00	TXN00000002	A00008149	PLC00009594	4/21/18	5/13/20	5/18/20	Property	141.71	
2	6/1/20 0:00	TXN00000003	A00003172	PLC00007969	10/3/19	5/21/20	5/26/20	Property	157.24	
3	6/1/20 0:00	TXN00000004	A00007572	PLC00009292	11/29/16	5/14/20	5/19/20	Health	172.87	

Visualiser la taille du dataset, analyser la structure et les types des variables

In [53]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 38 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   TXN_DATE_TIME             10000 non-null  object
 1   TRANSACTION_ID            10000 non-null  object
 2   CUSTOMER_ID               10000 non-null  object
 3   POLICY_NUMBER             10000 non-null  object
 4   POLICY_EFF_DT             10000 non-null  object
 5   LOSS_DT                   10000 non-null  object
 6   REPORT_DT                 10000 non-null  object
 7   INSURANCE_TYPE            10000 non-null  object
 8   PREMIUM_AMOUNT            10000 non-null  float64
 9   CLAIM_AMOUNT              10000 non-null  int64
 10  CUSTOMER_NAME             10000 non-null  object
```

```
  18  AGE                       10000 non-null  int64
  19  TENURE                    10000 non-null  int64
  20  EMPLOYMENT_STATUS         10000 non-null  object
  21  NO_OF_FAMILY_MEMBERS      10000 non-null  int64
  22  RISK_SEGMENTATION         10000 non-null  object
  23  HOUSE_TYPE                10000 non-null  object
  24  SOCIAL_CLASS              10000 non-null  object
  25  ROUTING_NUMBER            10000 non-null  int64
  26  ACCT_NUMBER               10000 non-null  object
  27  CUSTOMER_EDUCATION_LEVEL  9471 non-null   object
  28  CLAIM_STATUS              10000 non-null  object
  29  INCIDENT_SEVERITY         10000 non-null  object
  30  AUTHORITY_CONTACTED       8055 non-null   object
  31  ANY_INJURY                10000 non-null  int64
  32  POLICE_REPORT_AVAILABLE   10000 non-null  int64
  33  INCIDENT_STATE            10000 non-null  object
  34  INCIDENT_CITY             9954 non-null   object
  35  INCIDENT_HOUR_OF_THE_DAY  10000 non-null  int64
  36  AGENT_ID                  10000 non-null  object
  37  VENDOR_ID                 6755 non-null   object
dtypes: float64(1), int64(9), object(28)
memory usage: 2.9+ MB
```

La statistique descriptive des variables (Top, min max, écart-type, moyenne...)

In [61]:
```
#Statistque descriptive

df.describe(include="all")
```

Out[61]:

	TXN_DATE_TIME	TRANSACTION_ID	CUSTOMER_ID	POLICY_NUMBER	POLICY_EFF_DT	LOSS_DT	REPORT_DT	INSURANCE_TYPE	PREMIUM_AMOUNT
count	10000	10000	10000	10000	10000	10000	10000	10000	10000.000000
nique	395	10000	10000	10000	3306	414	409	6	NaN
top	1/25/21 0:00	TXN00009984	A00008012	PLC00004200	11/23/11	4/16/21	5/8/21	Property	NaN
freq	41	1	1	1	10	37	42	1692	NaN
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	88.508595
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	48.315874
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.000000
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	62.015000
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	88.835000
75%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	121.902500
max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	200.000000

Analyser les valeurs manquantes et incohérentes

```
In [62]: # les valeurs manquantes en pourcentage par variable

         (df.isnull().sum()/ len(df)) * 100
```

```
Out[62]: TXN_DATE_TIME              0.00
         TRANSACTION_ID             0.00
         CUSTOMER_ID                0.00
         POLICY_NUMBER              0.00
         POLICY_EFF_DT              0.00
         LOSS_DT                    0.00
         REPORT_DT                  0.00
         INSURANCE_TYPE             0.00
         PREMIUM_AMOUNT             0.00
         CLAIM_AMOUNT               0.00
         CUSTOMER_NAME              0.00
         ADDRESS_LINE1              0.00
         ADDRESS_LINE2             85.05
         CITY                       0.54
         STATE                      0.00
         POSTAL_CODE                0.00
         SSN                        0.00
         MARITAL_STATUS             0.00
         AGE                        0.00
         TENURE                     0.00
         EMPLOYMENT_STATUS          0.00
         NO_OF_FAMILY_MEMBERS       0.00
         RISK_SEGMENTATION          0.00
         HOUSE_TYPE                 0.00
         SOCIAL_CLASS               0.00
         ROUTING_NUMBER             0.00
         ACCT_NUMBER                0.00
         CUSTOMER_EDUCATION_LEVEL   5.29
         CLAIM_STATUS               0.00
         INCIDENT_SEVERITY          0.00
         AUTHORITY_CONTACTED       19.45
         ANY_INJURY                 0.00
         POLICE_REPORT_AVAILABLE    0.00
         INCIDENT_STATE             0.00
         INCIDENT_CITY              0.46
         INCIDENT_HOUR_OF_THE_DAY   0.00
         AGENT_ID                   0.00
         VENDOR_ID                 32.45
         dtype: float64
```

```
#Suppressions des colonnes inutiles

df_drop=df.drop(["POLICY_NUMBER", "ADDRESS_LINE1", "ADDRESS_LINE2","CITY","SSN","ROUTING_NUMBER","ACCT_NUMBER"], axis=1,inplace
```

```
In [66]: #Nombres de colonnes après suppression
         df.shape
```

```
Out[66]: (10000, 31)
```

Correction des valeurs manquantes des **variables catégorielles** par le **mode**

```
In [74]: # Remplacer toutes les valeurs manquantes des variables qualitatives par le mode

         for col in df.select_dtypes(include='object').columns:
             df[col].fillna(df[col].mode()[0])
```

```
In [75]: #Vérification d'existance des valeurs manquantes

         (df.isnull().sum()/ len(df)) * 100
```

Out[75]:

TXN_DATE_TIME	0.0
TRANSACTION_ID	0.0
CUSTOMER_ID	0.0
POLICY_EFF_DT	0.0
LOSS_DT	0.0
REPORT_DT	0.0
INSURANCE_TYPE	0.0
PREMIUM_AMOUNT	0.0
CLAIM_AMOUNT	0.0
CUSTOMER_NAME	0.0
STATE	0.0
POSTAL_CODE	0.0
MARITAL_STATUS	0.0
AGE	0.0
TENURE	0.0
EMPLOYMENT_STATUS	0.0
NO_OF_FAMILY_MEMBERS	0.0
RISK_SEGMENTATION	0.0
HOUSE_TYPE	0.0
SOCIAL_CLASS	0.0
CUSTOMER_EDUCATION_LEVEL	0.0
CLAIM_STATUS	0.0
CLAIM_STATUS	0.0
INCIDENT_SEVERITY	0.0
AUTHORITY_CONTACTED	0.0
ANY_INJURY	0.0
POLICE_REPORT_AVAILABLE	0.0
INCIDENT_STATE	0.0
INCIDENT_CITY	0.0
INCIDENT_HOUR_OF_THE_DAY	0.0
AGENT_ID	0.0
VENDOR_ID	0.0

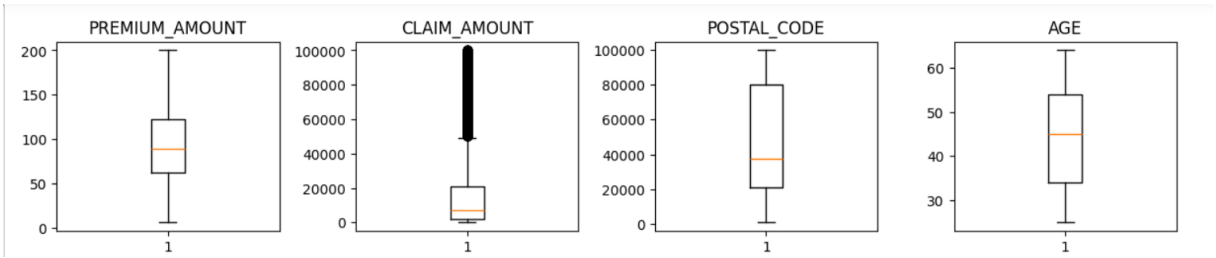
dtype: float64

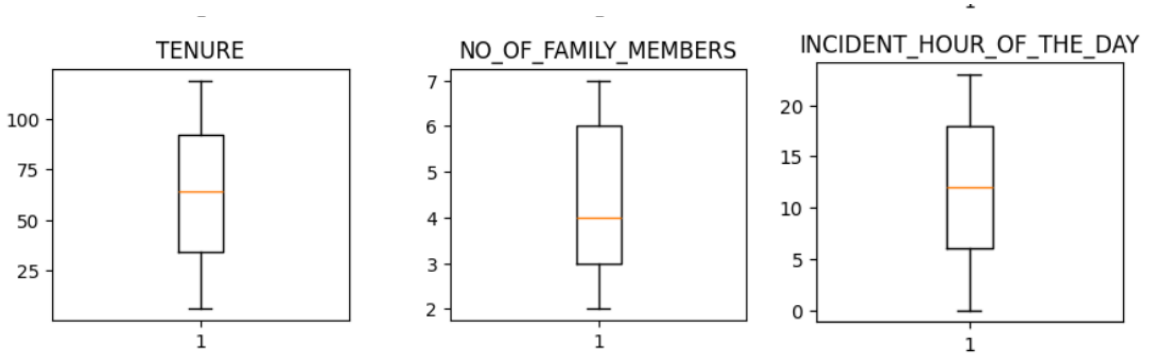
Analyse des **doublons** complets dans le dataset

In [79]:
```
# Compter les lignes dupliquées exactes
num_duplicates = df.duplicated().sum()
print("Nombre de doublons :", num_duplicates)
```

```
Nombre de doublons : 0
```

Analyse des valeurs aberrantes des **variables numériques** avec la **Boxplot**





NB : aucune Normalisation, car sous l'aspect métier, les valeurs aberrantes sont justifiées, vu que la période d'analyse des sinistres coïncide avec la pandémie de Covid 19

II. Chargement du dataset nettoyé sous python dans le flow

Process :

Créer un dataset vide (+) dans le flow :

Dataset managed => nom du dataset puis « create », ce nouveau dataset est alors créé dans le flow mais vide de l'intérieur.

Revenir dans le notebook python puis charger le dataset nettoyé dans le nouveau dataset créé dans le flow via un script python.

Objectif : gain de temps, nettoyage rapide et automatisé sous python du dataset.

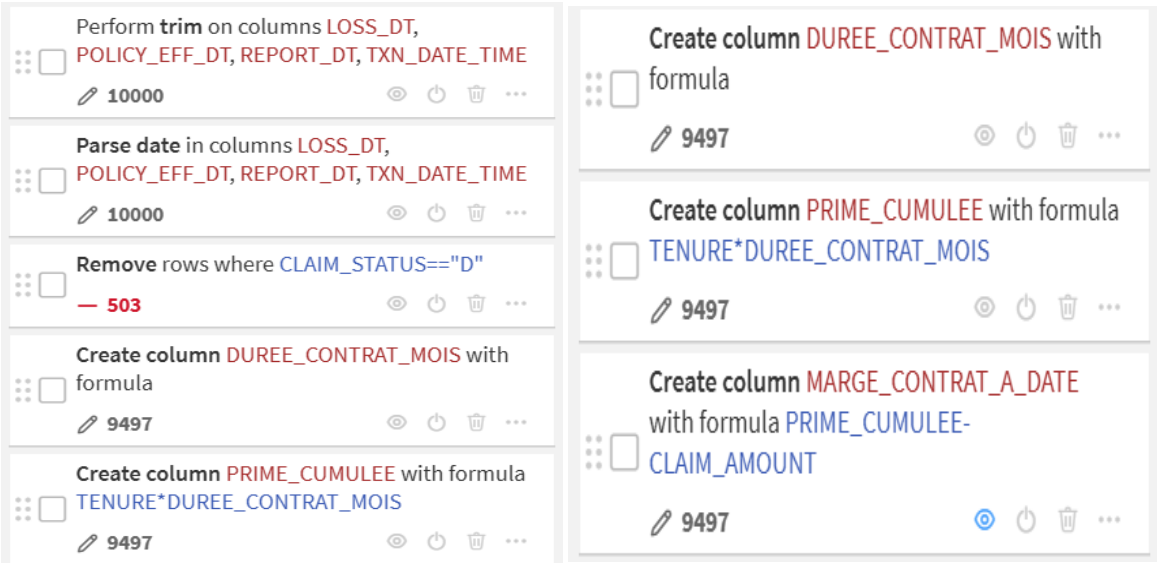
- ✓ Analyse supplémentaire du dataset clean dans le flow avec des recipes visuels

Recipe : **PREPARE**

- Analyse et correction du format sémantique (type donnée des variables dans Dataiku) : en bleu

- Suppression des vides « texte » entre les dates avec whitspace
- Parser les dates au format Dataiku (Parser date)
- Calculer/ créer des colonnes (Formule => math, diff(date)...)

NB : ci-dessous la capture d'écran de nettoyage et de transformations supplémentaires via la recette visuelle de dataiku (Prepare).



Recipe : JOIN

- JOINDRE les deux autres datasets au dataset principal nettoyé précédemment.
- **Left join** avec la clé ID_Vendor pour la table Vendor et ID_agents pour la table Agents.

(Pas de préfiltre, sortie dans une table annexe des lignes qui ne matchent pas, et non les supprimer)

- Analyse de la propriété du type de jointure
- Analyses de colonnes en sortie de la jointure

NB : le **left join** m'a permis de voir qu'il existe des vendeurs qui n'ont pas de clients ou qui ne leurs ont pas été attribués (questions métiers, sur ces vendeurs ou partenaires : à creuser).

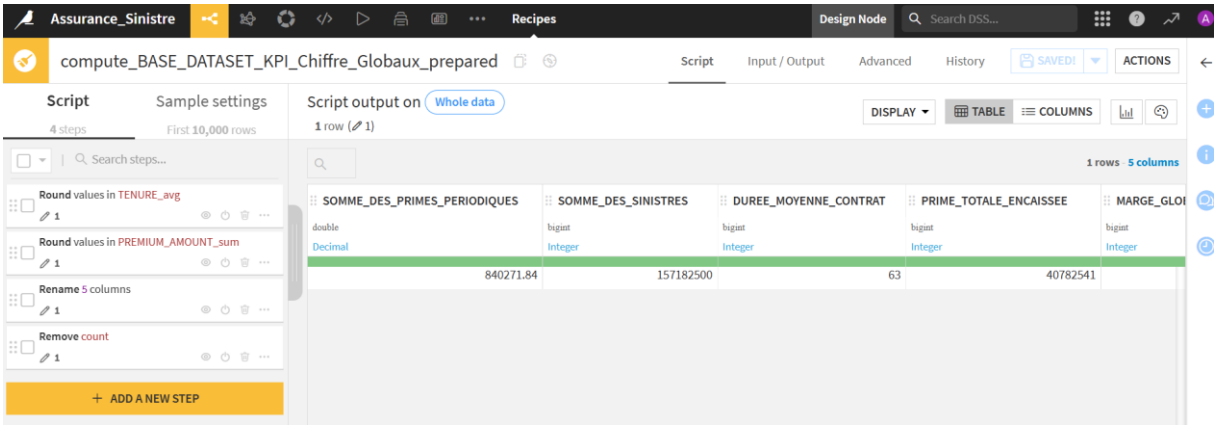
Recipe **: Group**

Pour une analyse rapide de quelques chiffres clés via des agrégations

(Mesures et dimensions).

54 rows · 8 columns

INSURANCE_...	SOCIAL_CL...	RISK_SEGMENTAT...	PREMIUM_AMOUN...	CLAIM_AMOUNT...	AGE_avg	MARGE_CONTRAT_A_DATE...	count
string	string	string	double	bigint	double	bigint	bigint
Text	Text	Text	Decimal	Integer	Decimal	Integer	Integer
Health	HI	H	9931.34	799000	44.0	-489792	68
Health	HI	L	25431.57	1845000	43.0	-1076199	172
Health	HI	M	21214.23	1586000	44.0	-967865	144
Health	LI	H	9496.67	685000	45.0	-420340	63
Health	LI	L	29021.77	2100000	43.0	-1235178	192



Flow illustratif ci-dessous :

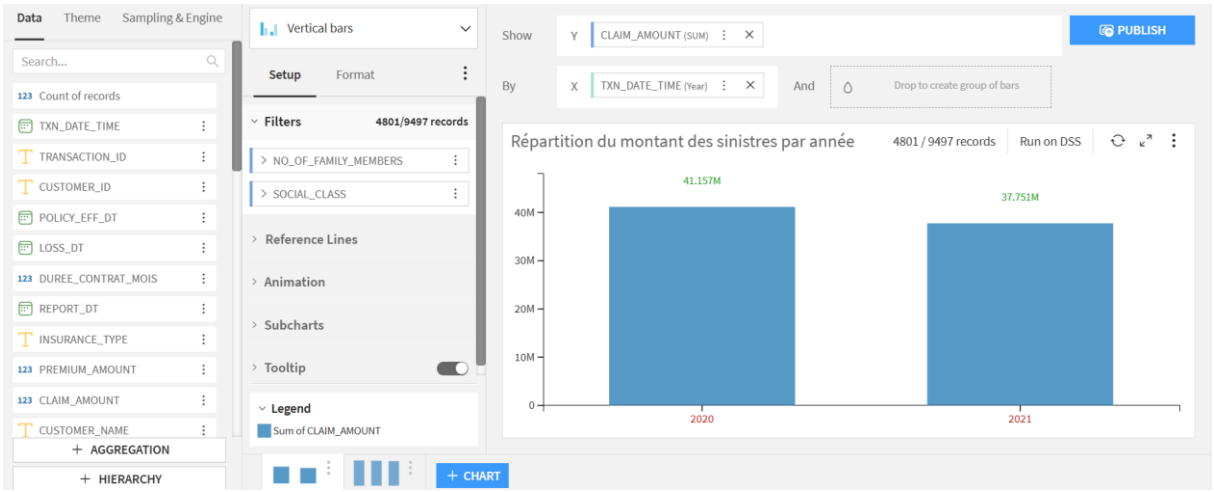
Dataset global nettoyé et modélisé.

III. Visualisation dataiku : Dashboard et KPIs

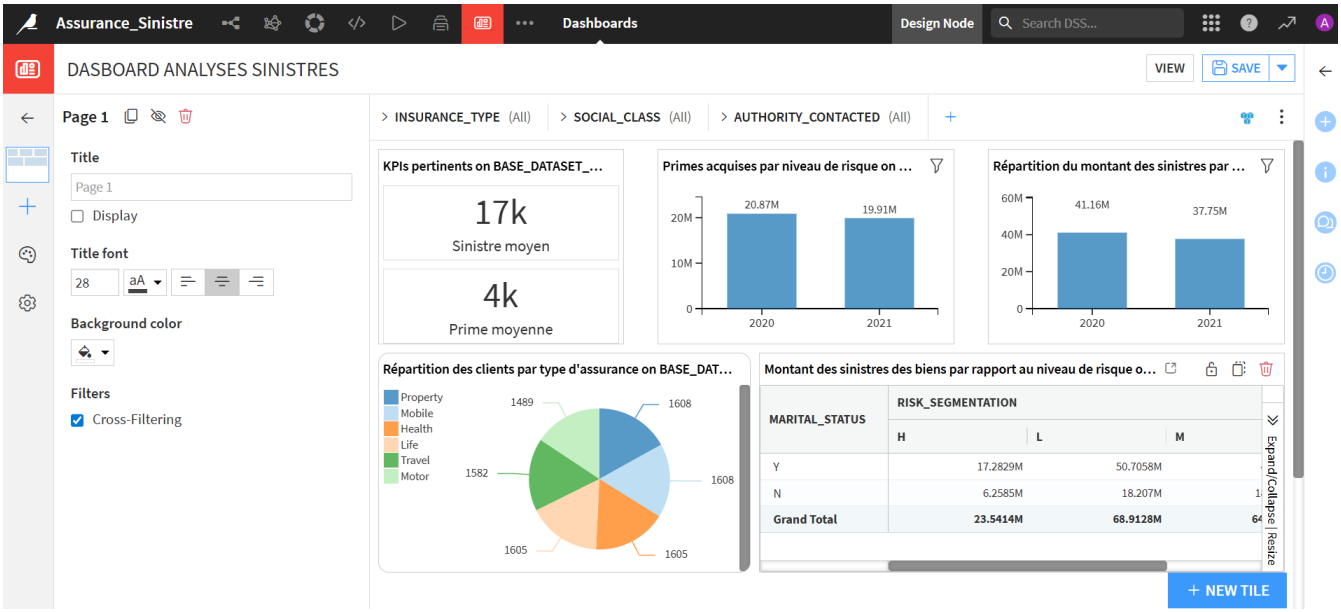
Les datasets étant consolidés en un dataset unique et nettoyé on va procéder à la création de métriques et de tableaux de bords dans Dataiku.

Process :

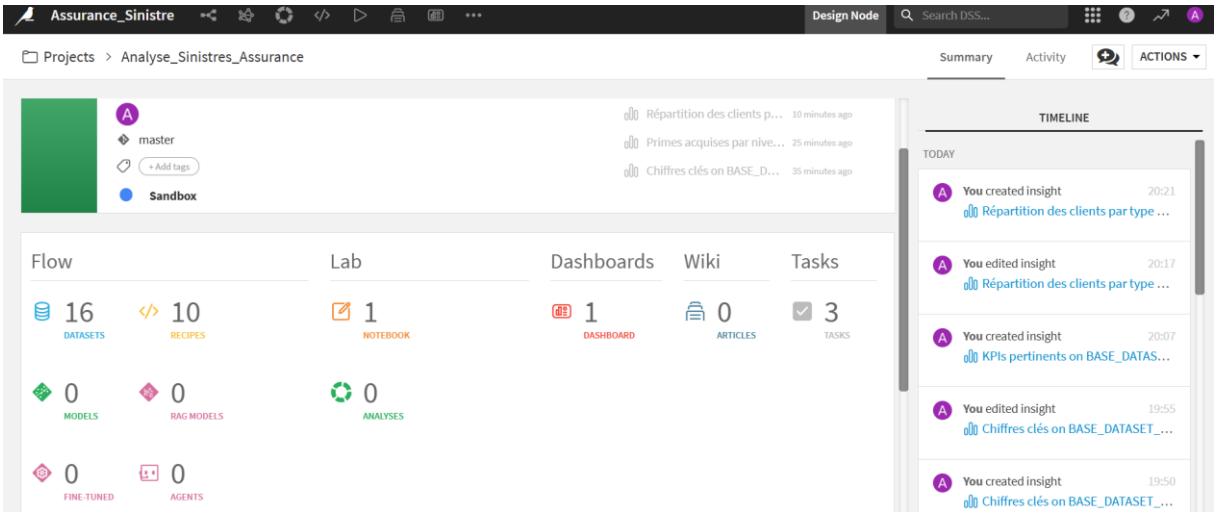
- Ouvrir le dataset final puis « CHART » : choisir le type de visualisation et commencer la constructions de ces visuels tout comme dans Power BI



- puis PUBLISH pour publier les visuels dans un Dashbord avec mise à jour automatique et une interactivité.
- Ajouter des filtres puis faire l'exportation.



Résumé historique (parcours) de la réalisation du projet en quelques point sur Dataiku :



Flow final des datasets et recettes utilisées :