Vizier DB - WebUser Interface Documentation

Release 1.0

New York University

May 15, 2019

Contents

Vizier is a new powerful tool to streamline the data curation process. Data curation (also known as data preparation, wrangling, or cleaning) is a critical stage in data science in which raw data is structured, validated, and repaired. Data validation and repair establish trust in analytical results, while appropriate structuring streamlines analytics.

Vizier makes it easier and faster to explore and analyze raw data by combining a simple notebook interface with spreadsheet views of your data. Powerful back-end tools that track changes, edits, and the effects of automation. These forms of provenance capture both parts of the exploratory curation process - how the cleaning workflows evolve, and how the data changes over time.

Vizier is a collaboration between the **University at Buffalo**, **New York University**, and the **Illinois Institute of Technology**.

CHAPTER 1

Contents

1.1 Install and Run

1.1.1 Use Docker Images

Install docker if it is not already: (install docker)[https://docs.docker.com/install/linux/docker-ce/debian/]

To run an instance of VizierDB using docker save the following script as run-containers.sh and make it executable:

```
#create network
sudo docker network create spark-net

#run the containers
#spark-master
MASTER_HOSTNAME="namenode"
MASTER_CONTAINER=`sudo docker run -d -v spark-data:/tmp --name $MASTER_HOSTNAME -h
↪$MASTER_HOSTNAME --network spark-net -p 222:22 -p 4040:4040 -p 6066:6066 -p␣
↪7077:7077 \
-p 8020:8020 -p 8080:8080 -p 50070:50070 --expose 7001 --expose 7002 --expose 7003 --
↪expose 7004 --expose 7005 --expose 7006 --expose 7077 --expose 6066 --expose 4040 \
--expose 8020 --expose 50070 -e "MASTER=spark://namenode:7077" -e "SPARK_CONF_DIR=/
↪conf" -e "SPARK_PUBLIC_DNS=127.0.0.1" -e "LD_LIBRARY_PATH=/usr/local/hadoop/lib/
↪native/" \
-e "SPARK_EXECUTOR_MEMORY=8g" -e "SPARK_DAEMON_MEMORY=8g" -e "SPARK_DRIVER_MEMORY=8g"␣
↪-e "SPARK_WORKER_MEMORY=8g" -e "HDFS_CONF_dfs_client_use_datanode_hostname=true" \
-e "AWS_ECS=false" docker.mimirdb.info/spark-hadoop /usr/local/spark-2.4.0-bin-
↪without-hadoop/master.sh`
echo "master container id: $MASTER_CONTAINER"
#wait for master to be ready
sleep 5

#spark-workers
START_PORT=7001
END_PORT=7006
```

(continues on next page)

(continued from previous page)

```
WORKER_PORT=8882
DATANODE_PORT=50010
#for additional spark workers increment the count below
SPARK_WORKERS_COUNT=2
i="0"
while [ $i -lt $SPARK_WORKERS_COUNT ]
do
        WORKER_WEBUI_PORT=$[$WORKER_WEBUI_PORT+$i]
        DATANODE_HOSTNAME="datanode$i"
        sudo docker run -d -v spark-data:/tmp -h $DATANODE_HOSTNAME --name $DATANODE_
↪HOSTNAME --network spark-net --link $MASTER_CONTAINER -p $WORKER_WEBUI_PORT:8082 \
        --expose $WORKER_PORT --expose $DATANODE_PORT -e "SPARK_CONF_DIR=/conf" -e
↪"SPARK_PUBLIC_DNS=127.0.0.1" -e "SPARK_WORKER_CORES=4" -e "SPARK_WORKER_PORT=
↪$WORKER_PORT" \
        -e "SPARK_WORKER_WEBUI_PORT=$WORKER_WEBUI_PORT" -e "LD_LIBRARY_PATH=/usr/
↪local/hadoop/lib/native/" -e "HDFS_DATA_HOST=$DATANODE_HOSTNAME" -e "HDFS_HOST=
↪$MASTER_HOSTNAME" \
        -e "HDFS_CONF_dfs_datanode_address=0.0.0.0:$DATANODE_PORT" -e "SPARK_EXECUTOR_
↪MEMORY=8g" -e "SPARK_DAEMON_MEMORY=8g" -e "SPARK_DRIVER_MEMORY=8g" -e "SPARK_WORKER_
↪MEMORY=8g" \
        -e "HDFS_CONF_dfs_client_use_datanode_hostname=true" -e "AWS_ECS=false"␣
↪docker.mimirdb.info/spark-hadoop /usr/local/spark-2.4.0-bin-without-hadoop/worker.sh
        i=$[$i+1]
done

VIZIER_DOMAIN="vizier.dev"

S3_AWS_ACCESS_KEY_ID="your_aws_access_key_id"
S3_AWS_SECRET_ACCESS_KEY="your_aws_secret"
S3_BUCKET_NAME="your_bucket_name"
VIZIER_DATA_VOLUME="vizier-data"

#mimir
#to use an s3 bucket as the data directory for mimir instead of a volume use this:
#sudo docker run -d -v mimir-data:/tmp/data/mimir -p 9002:9001 --expose 4041 --expose␣
↪33388 --network spark-net -h vizier-mimir --name vizier-mimir -e MIMIR_HOST="vizier-
↪mimir" \
#-e SPARK_HOST=$MASTER_HOSTNAME -e RESTORE_BACKUP=false -e PULL_MIMIR=false -e AWS_
↪ACCESS_KEY_ID=$S3_AWS_ACCESS_KEY_ID -e AWS_SECRET_ACCESS_KEY=$S3_AWS_SECRET_ACCESS_
↪KEY \
#-e S3_BUCKET_NAME="$S3_BUCKET_NAME" -e MIMIR_DATA_DIR="/tmp/data/mimir" --privileged␣
↪--device /dev/fuse docker.mimirdb.info/vizier-mimir-spark
#to use a local bind mount for the data directory instead of an s3 bucket use the␣
↪following for mimir instead of the above:
sudo docker run -d -v $VIZIER_DATA_VOLUME:/usr/local/source/web-api/.vizierdb -p␣
↪9002:9001 --expose 4041 --expose 33388 --network spark-net -h vizier-mimir --name␣
↪vizier-mimir \
-e USE_S3_VOLUME=false -e MIMIR_HOST="vizier-mimir" -e SPARK_HOST=$MASTER_HOSTNAME -e␣
↪RESTORE_BACKUP=false -e PULL_MIMIR=false -e AWS_ACCESS_KEY_ID=$S3_AWS_ACCESS_KEY_ID␣
↪\
-e AWS_SECRET_ACCESS_KEY=$S3_AWS_SECRET_ACCESS_KEY -e S3_BUCKET_NAME="$S3_BUCKET_NAME
↪" docker.mimirdb.info/vizier-mimir-spark

#api
#to use an s3 bucket as the data directory for the api instead of a volume use this:
#sudo docker run -d -p 9003:9001 --expose 80 --network spark-net -h vizier-api --name␣
↪vizier-api -e MIMIR_HOST="vizier-mimir" -e APP_PATH="" -e API_SERVER=api.$VIZIER_
↪DOMAIN \
```

(continues on next page)

(continued from previous page)

```
#-e API_LOCAL_PORT=80 -e API_PORT=443 -e API_SCHEME=https -e AWS_ACCESS_KEY_ID=$S3_
↪AWS_ACCESS_KEY_ID -e AWS_SECRET_ACCESS_KEY=$S3_AWS_SECRET_ACCESS_KEY \
#-e S3_BUCKET_NAME="$S3_BUCKET_NAME" --privileged --device /dev/fuse docker.mimirdb.
↪info/vizier-api-spark
#to use a local volume for the data directory instead of an s3 bucket use the␣
↪following for api instead of the above:
sudo docker run -d -v $VIZIER_DATA_VOLUME:/usr/local/source/web-api/.vizierdb -p␣
↪9003:9001 --expose 80 --network spark-net -h vizier-api --name vizier-api \
-e USE_S3_VOLUME=false -e MIMIR_HOST="vizier-mimir" -e MIMIR_URL="http://vizier-
↪mimir:8089/api/v2/" -e APP_PATH="/api" -e API_SERVER=demo.$VIZIER_DOMAIN \
-e API_LOCAL_PORT=80 -e API_PORT=443 -e API_SCHEME=https -e AWS_ACCESS_KEY_ID=$S3_AWS_
↪ACCESS_KEY_ID -e AWS_SECRET_ACCESS_KEY=$S3_AWS_SECRET_ACCESS_KEY \
-e S3_BUCKET_NAME="$S3_BUCKET_NAME" docker.mimirdb.info/vizier-api-spark

#ui
sudo docker run -d -e API_SERVER=demo.$VIZIER_DOMAIN -e APP_PATH="/api" -e API_
↪PORT=443 -e API_SCHEME=https \
--expose 80 --expose 443 -p 9004:9001 -h vizier-ui --name vizier-ui --network spark-
↪net docker.mimirdb.info/vizier-ui

#proxy
sudo docker run -d -p 80:80 -p 443:443 -p 9001:9001 -h vizier-proxy --name vizier-
↪proxy --network spark-net -e VIZIER_CONFIG="vizier_k8s.conf" \
-e VIZIER_API_APP_PATH="/api/" -e VIZIER_DOMAIN="$VIZIER_DOMAIN" -e VIZIER_API_PROXY_
↪PATH="/" docker.mimirdb.info/vizier-proxy
```

- update the VIZIER_DOMAIN variable for the vizier-proxy deployment to the domain you will use to access Vizier. You can use a real domain and DNS entries or the hosts file of a client. (Line 35)
- update the name or host paths for the volumes if you would like them somewhere other than the default (Line 40)
- update the s3-credentials and bucket name with your S3 access key id, secret, and bucket name: (Line 37, 38 39)

Run the script

```
>>> ./run-containers.sh
```

The IP address of the vizier-proxy service for a local docker deployment will likely be 127.0.0.1 After you have the IP of the vizier-proxy service you need to add the following entries to either DNS for a real domain or the hosts file of the client: so where vizier-proxy=127.0.0.1 and VIZIER_DOMAIN=vizier.dev

```
127.0.0.1 demo.vizier.dev
127.0.0.1 api.vizier.dev
```

Now you should be able to access the Vizier UI from a web browser.

```
https://demo.<VIZIER_DOMAIN>/vizier-db
```

1.1.2 Install Manually

Before installing Vizier DB Web UI, you should install VizierDB - Web API. The Web API is the backend that provides the API that is used by the Vizier DB Web UI.

1.1.3 Install VizierDB - Web API

Installation is still a bit labor intensive. The following steps seem to work for now (requires [Anaconda](https://conda.io/docs/user-guide/install/index.html)). If you want to use Mimir modules within your curation workflows a local installation of Mimir v0.2 is required. Refer to this [guide for Mimir installation details](https://github.com/VizierDB/Vistrails/tree/MimirPackage/vistrails/packages/mimir).

Python Environment

To setup the Python environment clone the repository and run the following commands:

```
>>> git clone https://github.com/VizierDB/web-api.git
>>> cd web-api
>>> conda env create -f environment.yml
>>> source activate vizier
>>> pip install git+https://github.com/VizierDB/Vistrails.git
>>> pip install -e .
```

As an alternative the following sequence of steps might also work (e.g., for MacOS):

```
>>> git clone https://github.com/VizierDB/web-api.git
>>> cd web-api
>>> conda create --name vizier pip
>>> source activate vizier
>>> pip install -r requirements.txt
>>> pip install -e .
>>> conda install pyqt=4.11.4=py27_4
```

Configuration

The web server is configured using a configuration file. There are two example configuration files in the (config directory)[https://github.com/VizierDB/web-api/tree/master/config] (depending on whether including Mimir `config-mimir.yaml` or not `config-default.yaml`). The configuration paramaters are:

api - *server_url*: Url of the server (e.g., http://localhost) - *server_port*: Server port (e.g., 5000) - *app_path*: Application path for Web API (e.g., /vizier-db/api/v1) - *app_base_url*: Concatenation of server_url, server_port and app_path - *doc_url*: Url to API documentation

fileserver - *directory*: Path to base directory for file server - *max_file_size*: Maximum size for file uploads

engines - *identifier*: Engine type (i.e., DEFAULT or MIMIR) - *name*: Engine printable name - *description*: Descriptive text for engine - *datastore*:

- directory: Base directory for data store

viztrails

- *directory*: Base directory for storing viztrail information and meta data

name: Web Service name

debug: Flag indicating whether server is started in debug mode

logs: Path to log directory

When the Web server starts it first looks for the configuration file that is reference in the environment variable `VIZIERSERVER_CONFIG`. If the variable is not set the server looks for a file `config.yaml` in the current working directory.

Note that there is a `config.yaml` file in the working directory of the server that can be used for development mode.

Run Server

After adjusting the server configuration the server is run using the following command:

```
>>> cd vizier
>>> python server.py
```

Make sure that the conda environment has been activated using `source activate vizier`.

If using Mimir the gateway server sould be started before running the web server.

API Documentation

For development it can be helpful to have a local copy of the API documentation. The [repository README](https://github.com/VizierDB/webapi-swagger-ui) contains information on how to install the UI locally.

1.1.4 Install VizierDB - Web UI

Start by cloning the repository and switching to the app directory.

```
>>> git clone https://github.com/VizierDB/web-ui.git
>>> cd web-ui
```

Inside the app directory, you can run several commands:

Install build dependencies

```
>>> yarn install
```

Start the development server

```
>>> yarn start
```

Bundles the app into static files for production

```
>>> yarn build
```

Additional Commands

Starts the test runner.

```
>>> yarn test
```

Remove this tool and copies build dependencies, configuration files and scripts into the app directory. If you do this, you can't go back!

```
>>> yarn eject
```

Configuration

The UI app connects to the Web API server. The Url for the server is currently hard-coded in the file `public/env.js`. Before running `yarn start` adjust the Url to point to a running Web API server. By default a local server running on port 5000 is used.

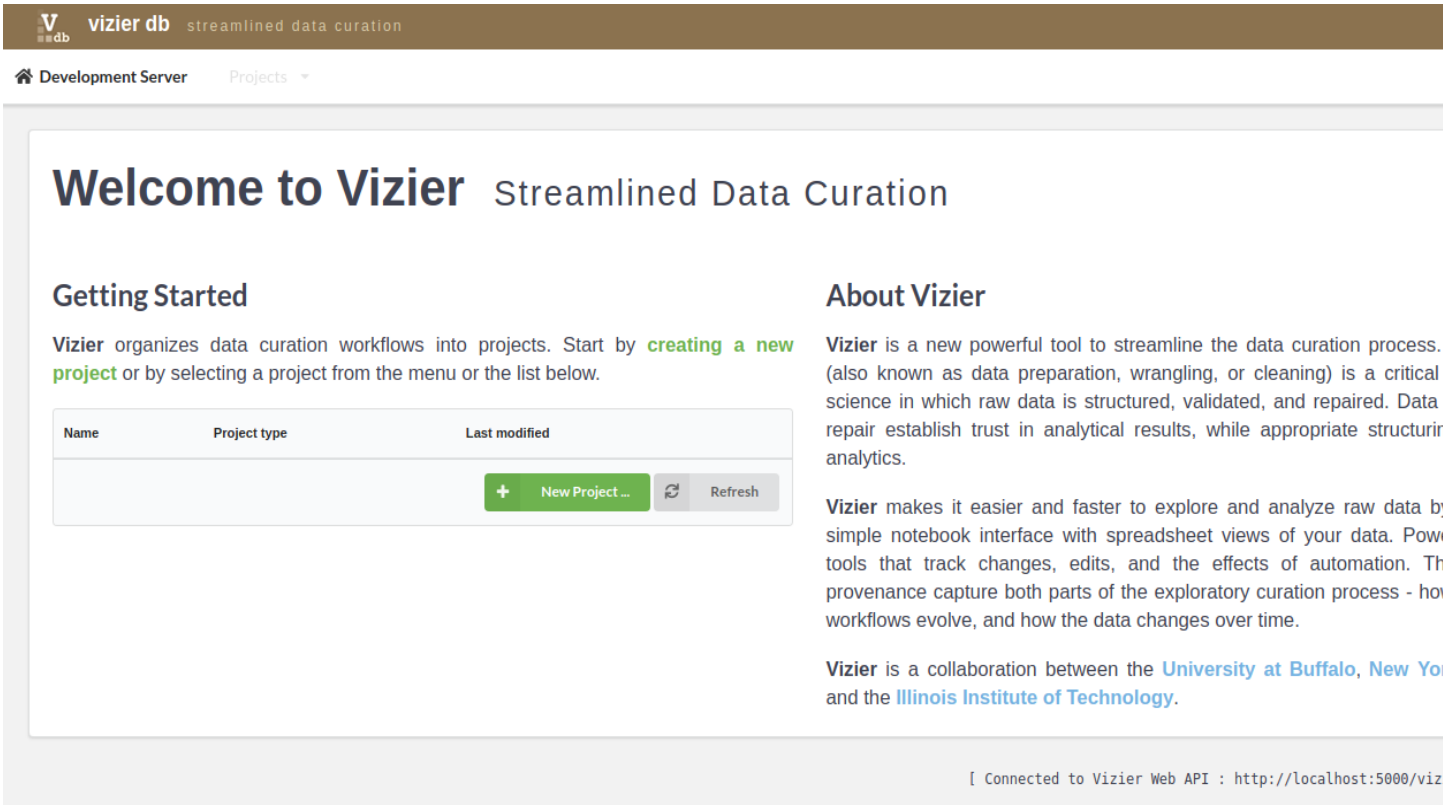
1.2 Getting Started

Vizier organizes data curation workflows into projects.

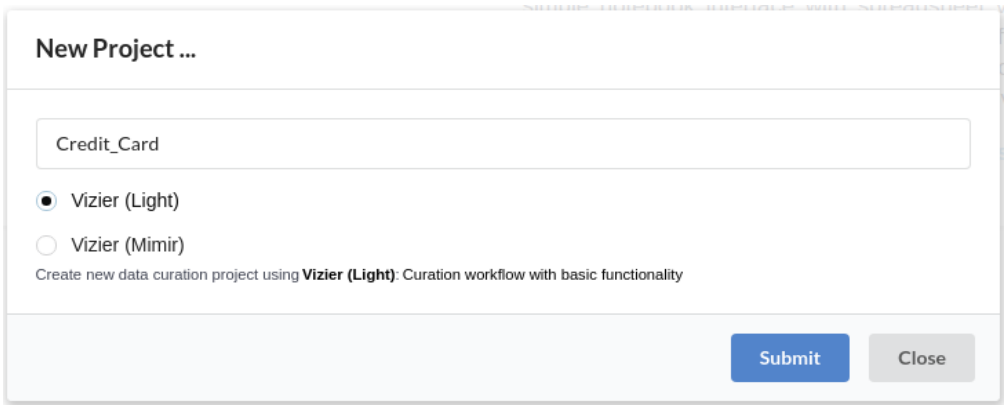
- Start by selecting or creating a new project under the Projects Tab.
- If the data that you want to clean is currently stored in CSV files, these files have to be uploaded to the file server. You can upload your data files under the Files Tab.

1.2.1 Step 1

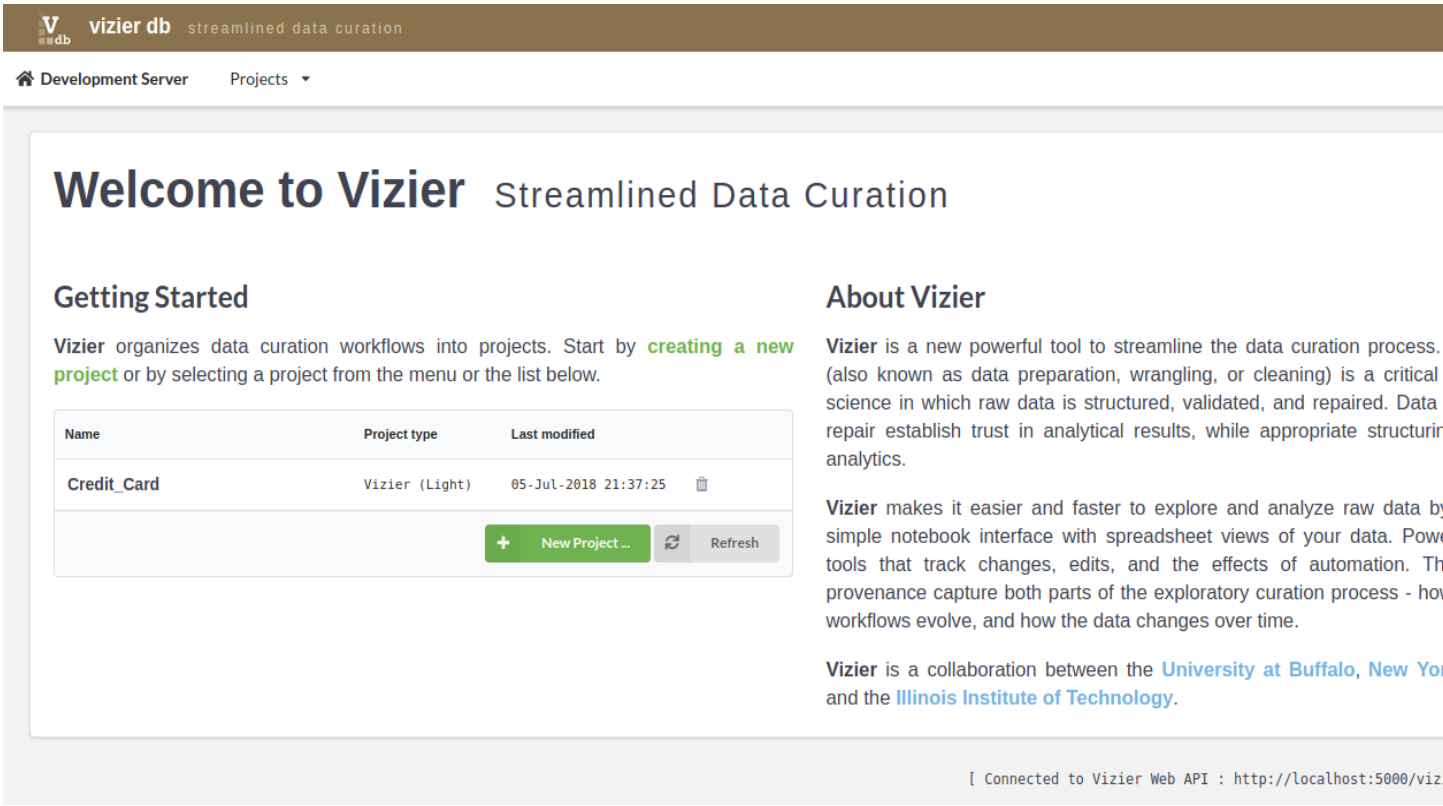
Create Project



Begin by adding a project on the Vizier page (initial page), shown in the figure above, by clicking on the **New Projects ...** button.



On the New Project... dialog shown in figure above, enter the name of the project you would like to create, for example **credit_card**, and click on **Submit** button. You should now see the new project you added in the list of projects as shown below.



Once project is added click on project name in the list of projects to data curation.

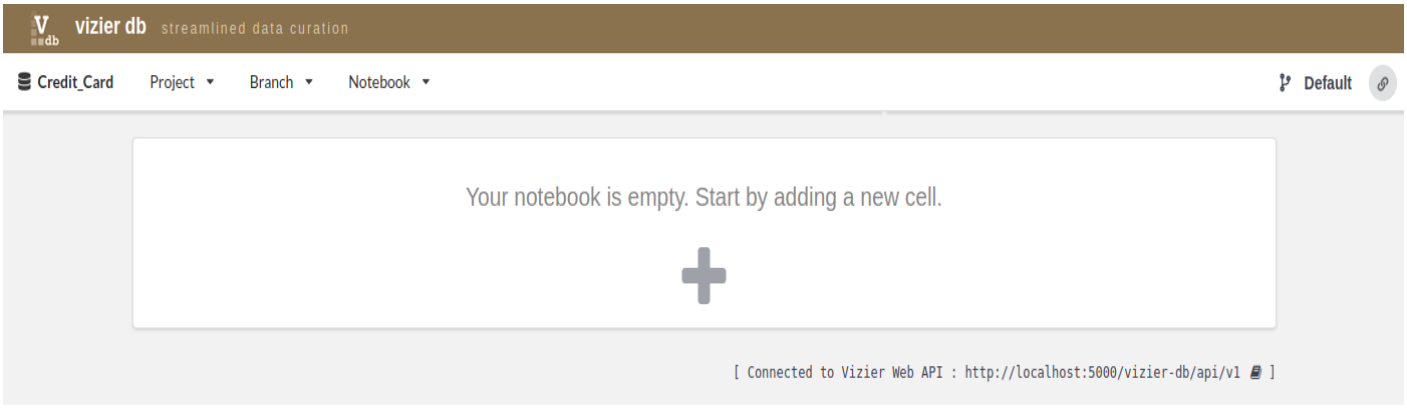
1.2.2 Step 2

Load Dataset

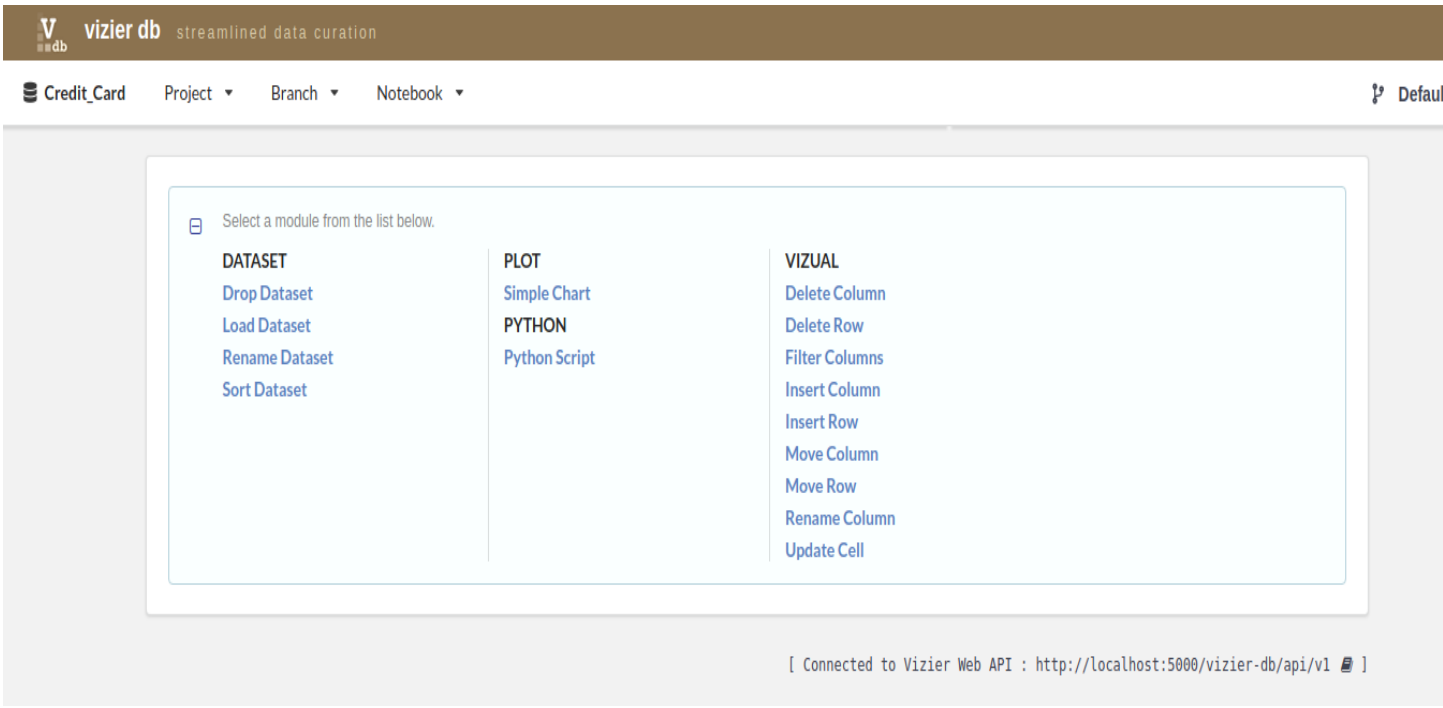
Continuing with our example of the **Credit_Card** project, we show here the methods of uploading data.

First, select one project from the list of projects, for example, **credit_card** project by clicking on the name project.

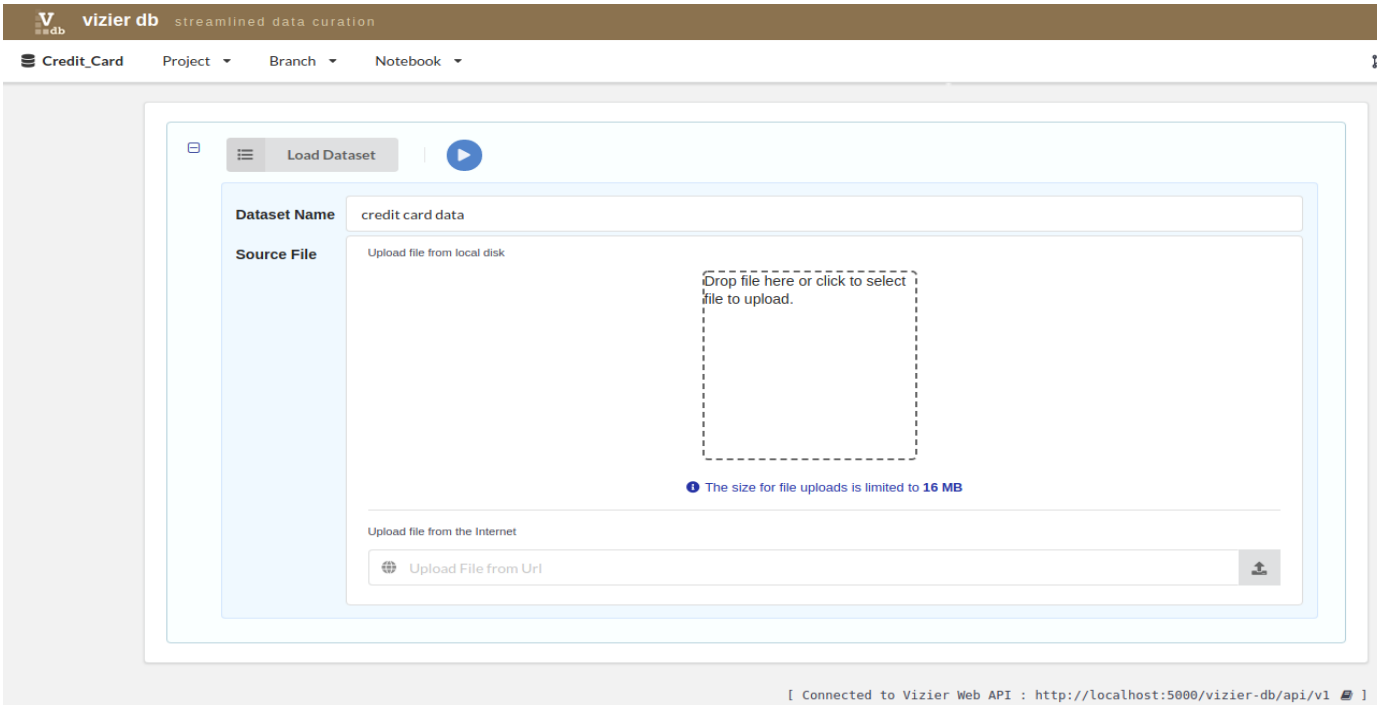
Once you are inside the project, load the data by clicking in the sign **+**.



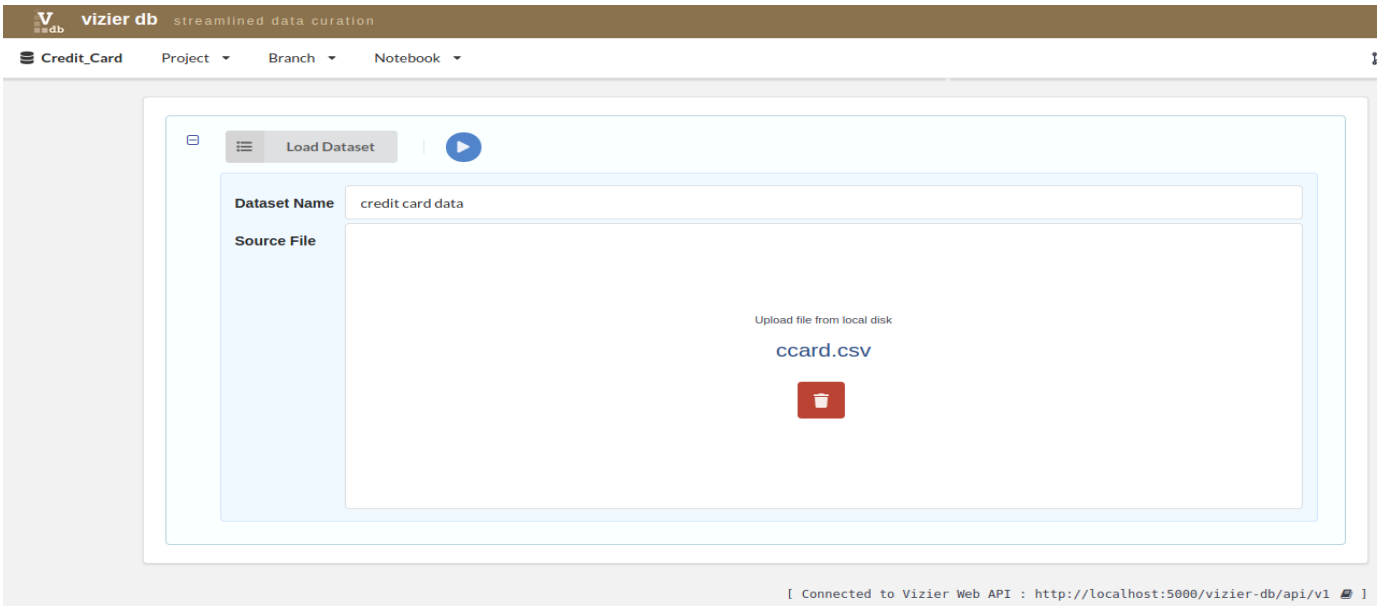
Then, go to the column **DATASET**, and click on **Load Dataset**



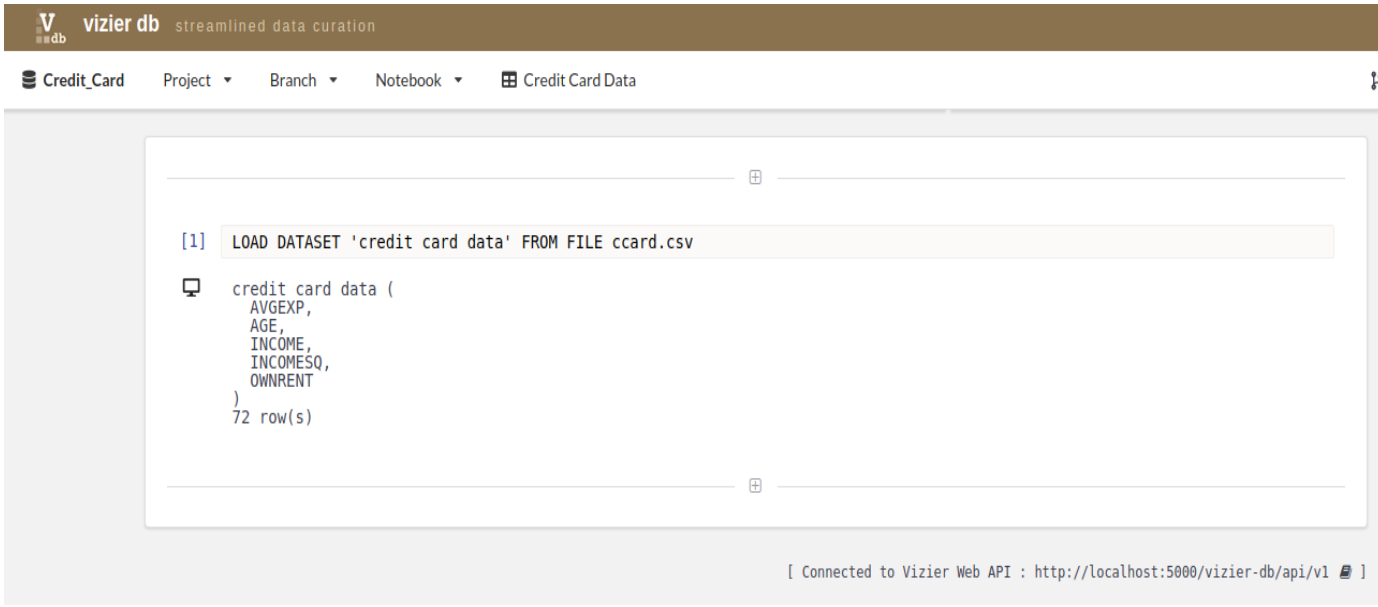
Then, upload the data set. You have to provide the data set name and the source file.



For example, we entered **credit card data** as the name of the dataset for that project and selected ccard.csv dataset, then, click on the blue **play** icon.



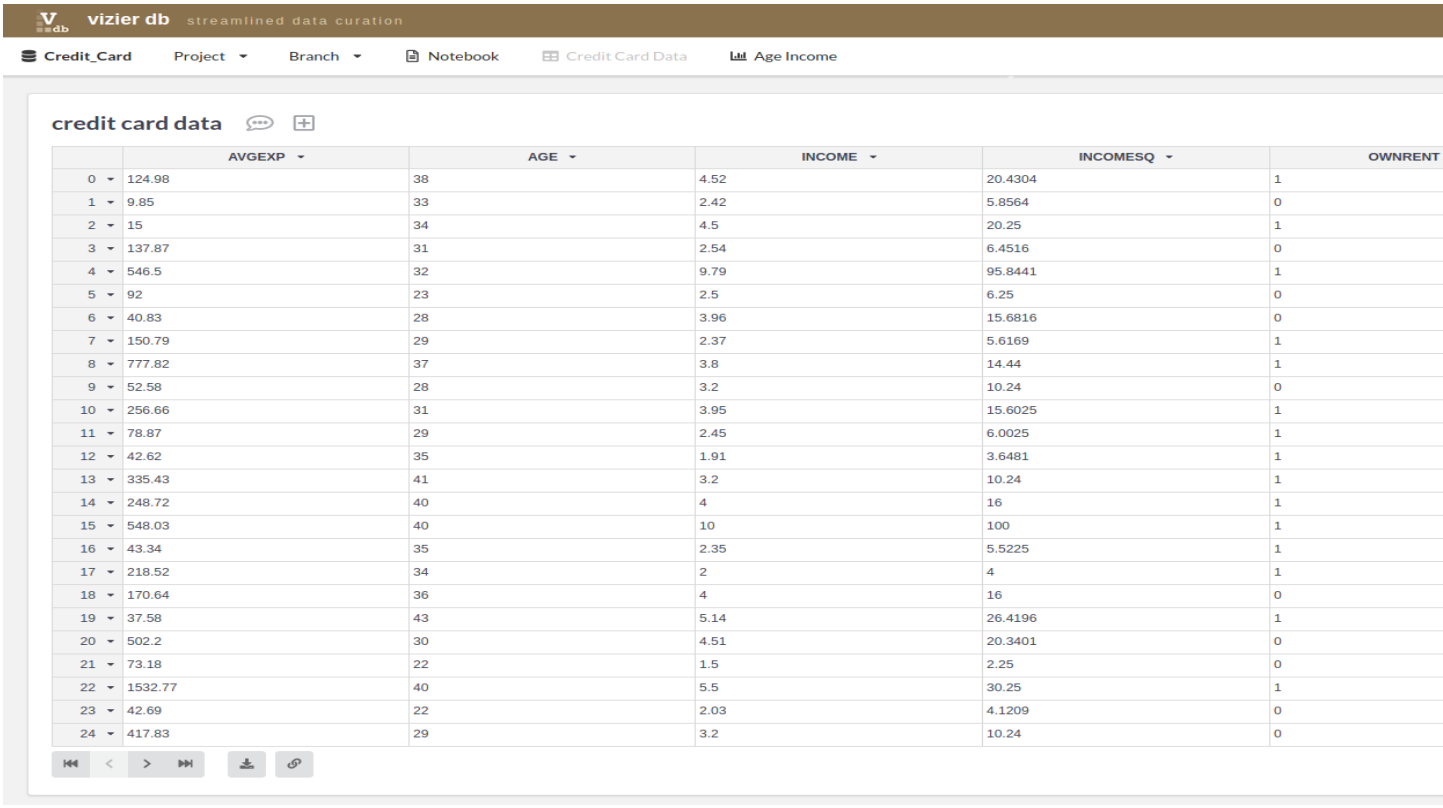
After loading the **credit card dataset**, we can start to explore and curate our data.



1.2.3 Step 3

Spreadsheet Views

Vizier makes it easier and faster to explore and analyze raw data by combining a simple notebook interface with spreadsheet views of your data. Powerful back-end tools that track changes, edits, and the effects of automation. These forms of provenance capture both parts of the exploratory curation process - how the cleaning workflows evolve, and how the data changes over time. To access the spreadsheet of our Credit Card project just go under the **Credit Card Data** Tab.

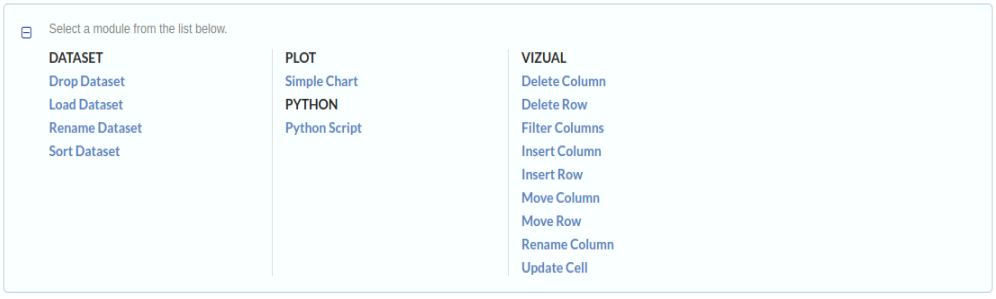


	AVGEXP	AGE	INCOME	INCOMESQ	OWNRENT
0	124.98	38	4.52	20.4304	1
1	9.85	33	2.42	5.8564	0
2	15	34	4.5	20.25	1
3	137.87	31	2.54	6.4516	0
4	546.5	32	9.79	95.8441	1
5	92	23	2.5	6.25	0
6	40.83	28	3.96	15.6816	0
7	150.79	29	2.37	5.6169	1
8	777.82	37	3.8	14.44	1
9	52.58	28	3.2	10.24	0
10	256.66	31	3.95	15.6025	1
11	78.87	29	2.45	6.0025	1
12	42.62	35	1.91	3.6481	1
13	335.43	41	3.2	10.24	1
14	248.72	40	4	16	1
15	548.03	40	10	100	1
16	43.34	35	2.35	5.5225	1
17	218.52	34	2	4	1
18	170.64	36	4	16	0
19	37.58	43	5.14	26.4196	1
20	502.2	30	4.51	20.3401	0
21	73.18	22	1.5	2.25	0
22	1532.77	40	5.5	30.25	1
23	42.69	22	2.03	4.1209	0
24	417.83	29	3.2	10.24	0

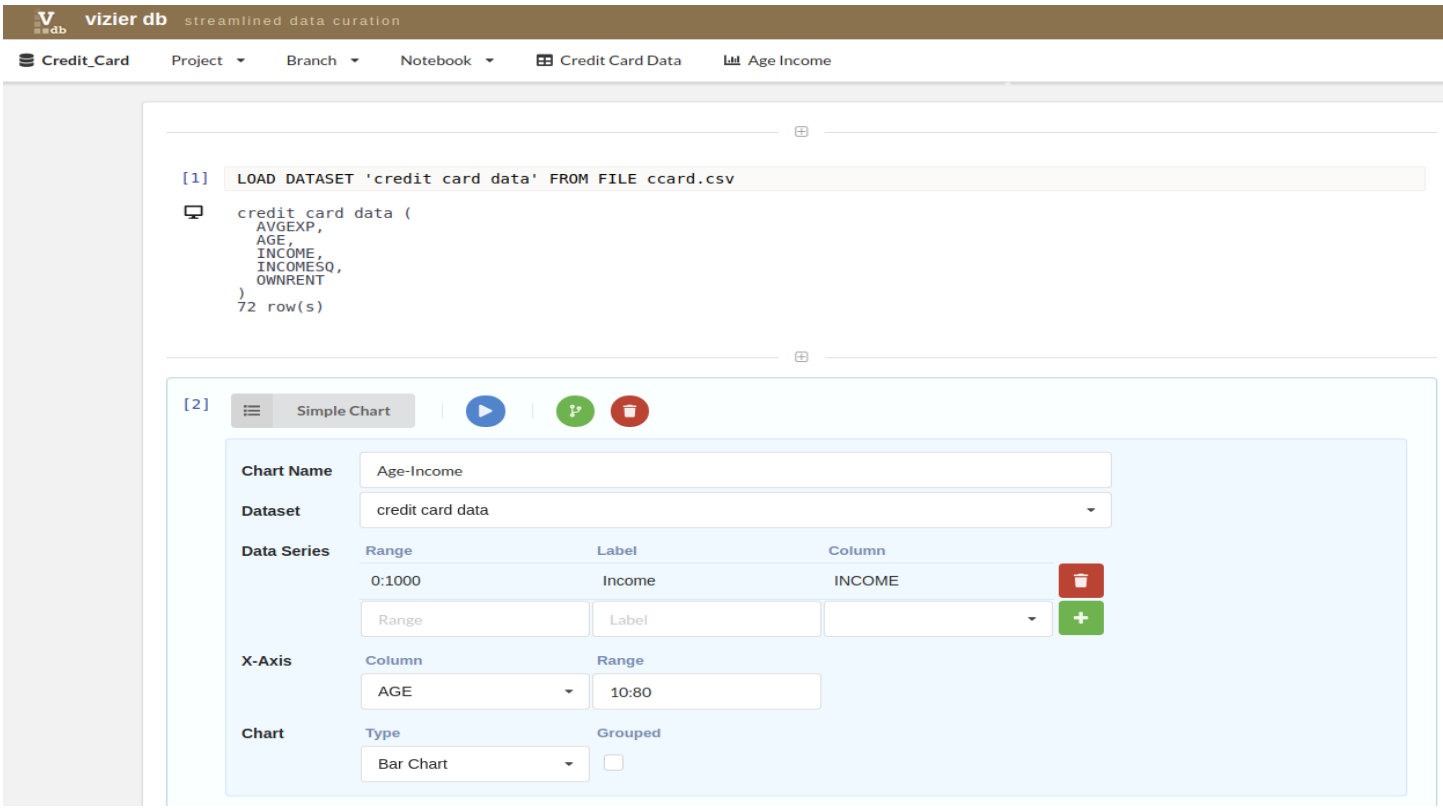
1.2.4 Step 4

Chart Views

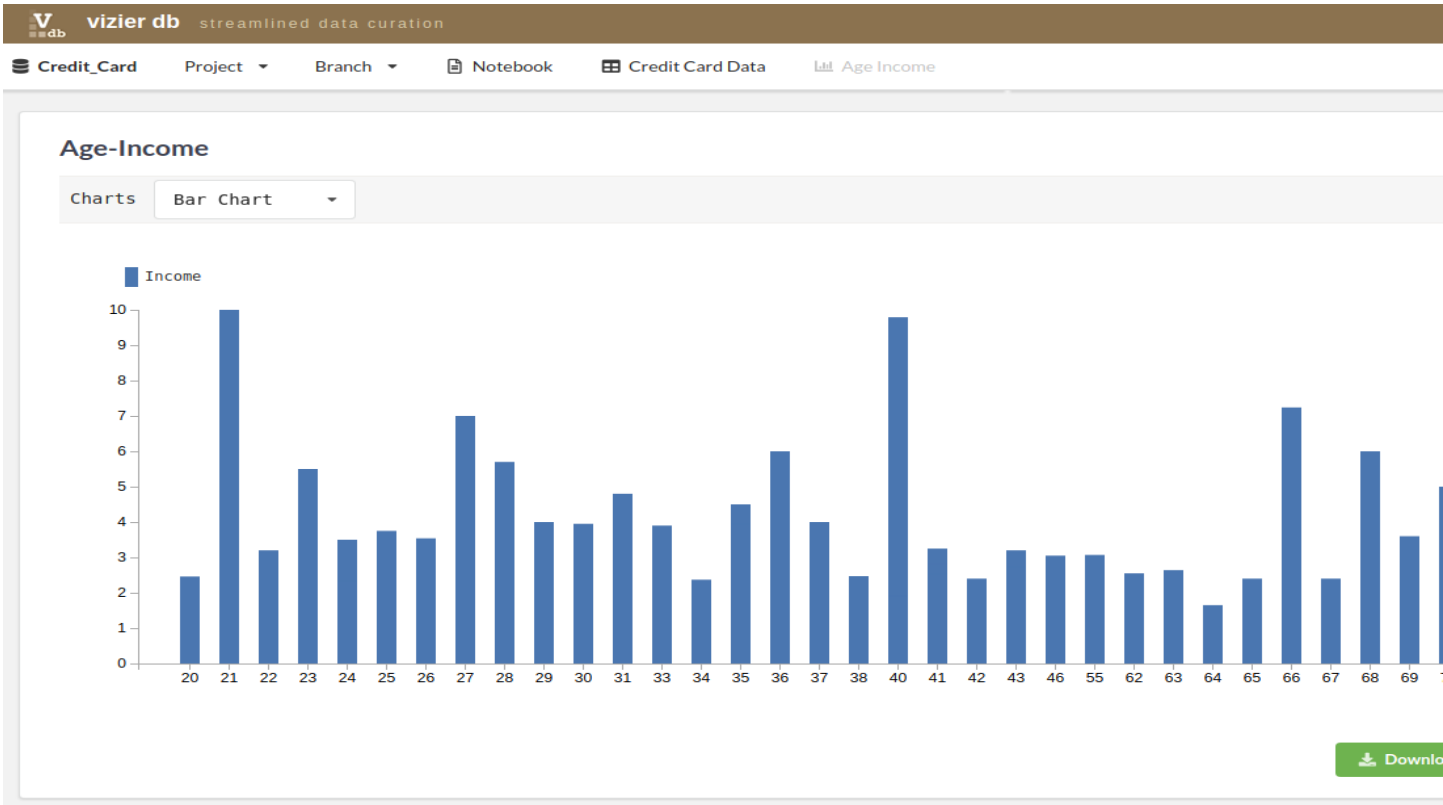
Vizier provides five types of chart: Simple bar chart, group bar chart, line chart, area chart and scatter plot. To create a chart, user have to select a module **Plot>>Simple Chart** from the list below.



Then, fill the form and click on the blue **play** icon.



To access the Chart view of our Credit Card project just go under the **Age Income** Tab which is the name of the chart.



CHAPTER 2

Links

- GitHub repository

CHAPTER 3

Indices and tables

- genindex
- modindex
- search