LA Crime Emulation-Based system

Fengsheng Zhou	Jesse(Yingkai)Zhong	Haodong Feng
6868360913	**6970547079**	**5416226236**
fengshen@usc.edu	yingkaiz@usc.edu	haodongf@usc.edu

Background

Nowadays, security has always been one of top issues on people's minds. When facing a choice related to location, the crime situation is taken into account without doubt. We believe that there are certain relations between crime situations and GDP or personal income in different areas. We hope to combine these information in order to create a platform that can provide security reference for people.

Dataset for the Project

GDP by County, Metro, and Other Areas:
https://www.bea.gov/data/gdp/gdp-county-metro-and-other-areas

Crime in the United States Annual Reports: (update by county)
https://ucr.fbi.gov/crime-in-the-u.s

Personal income Data by County:
https://www.bea.gov/data

In our project, there will be three datasets used. They are Crime data, GDP data and income data. Crime data stands for Crime in the United States Annual Reports by county. GDP data stands for GDP by County, Metro, and Other Areas. Income data stands for per capita Personal Income by County in the US.

Data Cleansing Process

Offenses Known to Law Enforcement by State by Metropolitan and Nonmetropolitan Counties, 2019 [The data shown in this table do not reflect county totals but are the number of offenses reported by the sheriff's office or county police department.]											
State	County	**Violent crime**	Murder and nonnegligent manslaughter	Rape[1]	Robbery	Aggravated assault	**Property crime**	Burglary	Larceny-theft	Motor vehicle theft	Arson[2]
ALABAMA[3]											
ARIZONA - Metropolitan Counties	Cochise	47	0	1	3	43	531	195	273	63	5
	Coconino	127	6	17	1	103	264	59	180	25	6
	Mohave	135	3	6	19	107	1,983	573	1,197	213	13
	Pinal	197	3	5	18	171	1,469	321	987	161	
	Yavapai	193	5	0	4	184	860	155	608	97	10
	Yuma	136	7	17	6	106	807	223	502	82	7
ARIZONA - Nonmetropolitan Counties	Gila	176	1	3	5	167	384	65	252	67	
	Graham	61	4	0	0	57	104	20	76	8	12
	La Paz	40	1	2	3	34	310	87	181	42	1
	Navajo	43	1	0	5	37	341	192	115	34	5
	Santa Cruz	1	0	0	0	1	117	35	66	16	0

The screenshot above is a representation of one of our three datasets, the original crime data dataset. We can see that in the original dataset, there are twelve columns to represent different information, such as State and County. Location information is the most important data in our research projects. The other ten columns are the specific information of the data. Violent crime and Property crime divide the major types of crime and case statistics, and the remaining eight columns show more specific crime types under the two major crime types. We know this data cannot be used directly. As we all know, raw data always comes in a form that isn't ready for analysis due to structural characteristics or even the quality of the data. Data cleaning and preparation is so important. As we can see below, after data cleansing, we converted the unstructured data into structured data. Each line will have sufficient information to indicate its state and county information. This is just a microcosm of a small part of the task of data cleaning. For the other two datasets GDP and income of the project, some of the data have missing values and NA values. We can take a look at how many missing values are in the data with the Pandas df.info method. To handle the missing values, We have two solutions. For rows with partially missing data, we use fillna to replace null values with zero. For rows with lots of missing data, we drop rows with 2 or more missing values using the dropna function. In the income dataset, both state and county information are aggregated in one column. We inserted a new column to make the information neater and easier to categorize. After that step, we use the reset index function after dropping rows & adding columns.

crime

	State	County	Violent_Crime	Murder	Rape	Robbery	Aggravated_Assault	Property_crime	Burglary	Larceny-theft	Motor_vehicle_theft	Arson
0	ARIZONA	Cochise	47.0	0.0	1.0	3.0	43.0	531.0	195.0	273.0	63.0	5.0
1	ARIZONA	Coconino	127.0	6.0	17.0	1.0	103.0	264.0	59.0	180.0	25.0	6.0
2	ARIZONA	Mohave	135.0	3.0	6.0	19.0	107.0	1983.0	573.0	1197.0	213.0	13.0
3	ARIZONA	Pinal	197.0	3.0	5.0	18.0	171.0	1469.0	321.0	987.0	161.0	0.0
4	ARIZONA	Yavapai	193.0	5.0	0.0	4.0	184.0	860.0	155.0	608.0	97.0	10.0
5	ARIZONA	Yuma	136.0	7.0	17.0	6.0	106.0	807.0	223.0	502.0	82.0	7.0
6	ARIZONA	Gila	176.0	1.0	3.0	5.0	167.0	384.0	65.0	252.0	67.0	0.0
7	ARIZONA	Graham	61.0	4.0	0.0	0.0	57.0	104.0	20.0	76.0	8.0	12.0
8	ARIZONA	La Paz	40.0	1.0	2.0	3.0	34.0	310.0	87.0	181.0	42.0	1.0
9	ARIZONA	Navajo	43.0	1.0	0.0	5.0	37.0	341.0	192.0	115.0	34.0	5.0
10	ARIZONA	Santa Cruz	1.0	0.0	0.0	0.0	1.0	117.0	35.0	66.0	16.0	0.0

Task 1.1: Build an EDFS use Firebase

We all know about Hadoop, a large-scale distributed & parallel batch- processing infrastructure. And same for HDFS(Hadoop distributed file system) which can distribute data storage with high reliability. In our project, we will build an emulated distributed file system which is a replica of HDFS . To achieve this function, we choose to define a lot of functions to implement HDFS commands in our project.

Create a Directory: mkdir()

Use json.loads() takes in data as string format and returns a json object. Then use json.dumps() to make sure the data is string format. Use requests.put() to send the new directory to the server to update and modify the content of the data, thereby changing the information.

```
mkdir("/user")
```

```
https://projec-549db-default-rtdb.firebaseio.com/
    user: "{}"
```

```
mkdir("/user/john")
```

```
https://projec-549db-default-rtdb.firebaseio.com/
    user
        john: "{}"
```

List all the files: ls()

Use requests.get() send a request to the database for data in the specified directory, then convert the json file to a dictionary and find the lists containing all the keys of the dictionary.

```
ls("/user")
```

```
john
```

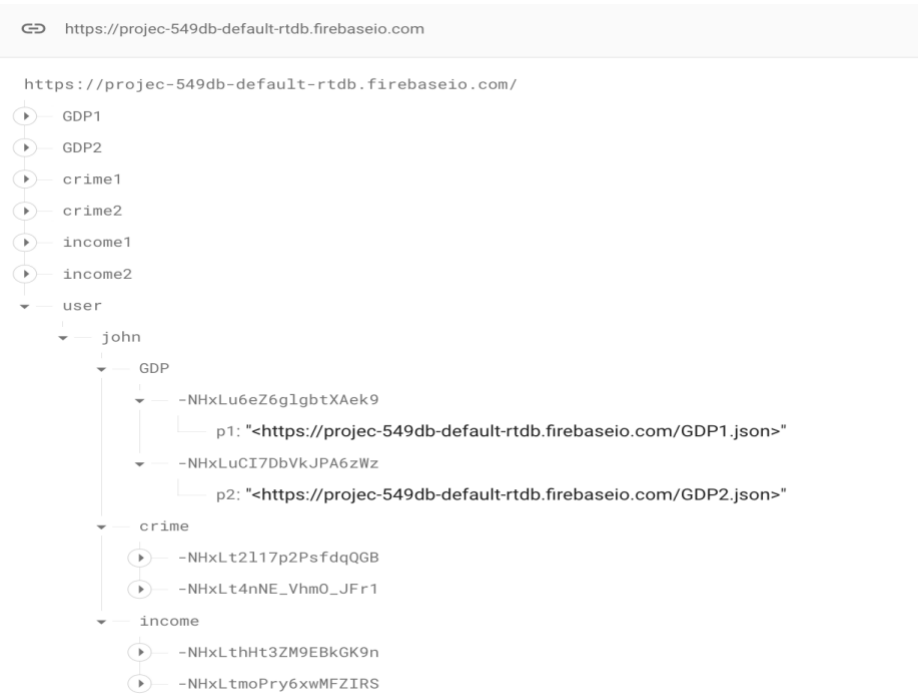
Uploading a file to file system: put()

Unlike the previous ones, because to create new content and change the data type, use the requests.post function according to the value of k (# partitions). Next, find how many rows that data has and do the equal split. In order to achieve equal split, we set min_rows and max_rows through the number of rows whether they are divisible by (partition +1) or (partition +3), return the floor division result. Then we use the random.randint function to generate random numbers in the range of min_rows and max_rows, stopping when the random number is less than or equal to 0. Then we read the csv file with the number of rows just generated, and as we did above save and write these dataframe to the csv file. Finally, we use request.put to update the data we post at

first. In that case, we can achieve the command and upload a file to the file system with k partitions.

```
put("crime.csv","/user/john",2)
```

```
put("income.csv","/user/john",2)
put("GDP.csv","/user/john",2)
```



Return the locations of partitions of the file: getPartitionLocations()

Use requests.get to send a request to the database for data about the locations of partitions, then convert the json file to a dictionary and find all the values available in the given dictionary. Then use for-loop to summarize the obtained information.

```
getPartitionLocations("/user/john/crime.csv")
```

```
['<https://projec-549db-default-rtdb.firebaseio.com/crime1.json>',
 '<https://projec-549db-default-rtdb.firebaseio.com/crime2.json>']
```

Return the content of partition # of the specified file: readPartition()

Write out the partition_location according to the partition# of the file, then use requests.get function send a request to the database for that specific data. Then get the json string using the json.dumps function.

```
readPartition("/user/john/crime.csv",1)
```

```
[{'Aggravated_Assault': 103.0,
  'Arson': 6.0,
  'Burglary': 59.0,
  'County': 'Coconino',
  'Larceny-theft': 180.0,
  'Motor_vehicle_theft': 25.0,
  'Murder': 6.0,
  'Property_crime': 264.0,
  'Rape': 17.0,
  'Robbery': 1.0,
  'State': 'ARIZONA',
  'Violent_Crime': 127.0,
  'id': 1},
 {'Aggravated_Assault': 107.0,
  'Arson': 13.0,
  'Burglary': 573.0,
  'County': 'Mohave',
  'Larceny-theft': 1197.0,
  'Motor_vehicle_theft': 213.0,
  'Murder': 3.0,
```

Display content of a file: cat()

Use requests.get to send a request to the database for data, like before we did. We convert the json file to a dictionary and find all the values available in the dictionary. However, this time we save the values. Next step is the same as we did in readPartition, download the new data and print it out as json string format.

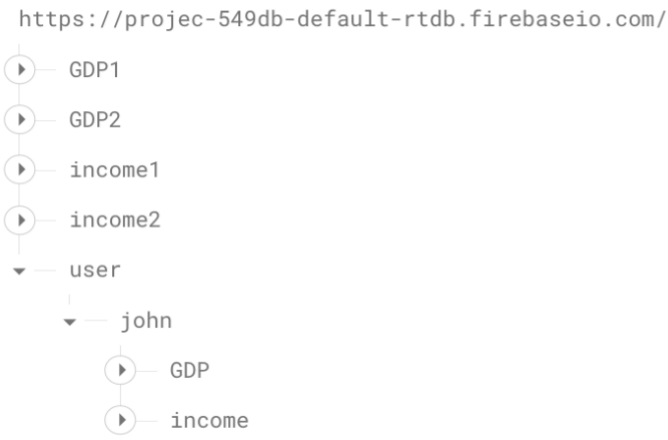
```
cat("/user/john/crime.csv")
```

```
partition from https://projec-549db-default-rtdb.firebaseio.com/crime1.json
[
  {
    "Aggravated_Assault": 103.0,
    "Arson": 6.0,
    "Burglary": 59.0,
    "County": "Coconino",
    "Larceny-theft": 180.0,
    "Motor_vehicle_theft": 25.0,
    "Murder": 6.0,
    "Property_crime": 264.0,
    "Rape": 17.0,
    "Robbery": 1.0,
    "State": "ARIZONA",
    "Violent_Crime": 127.0,
    "id": 1
  },
  {
    "Aggravated_Assault": 107.0,
    "Arson": 13.0,
```

Remove a file from the file system: rm()

Use requests.get to get the data from the database, like we did in cat function, save the url and use requests.delete function to delete the file we don't need anymore.

```
rm("/user/john/crime.csv")
```



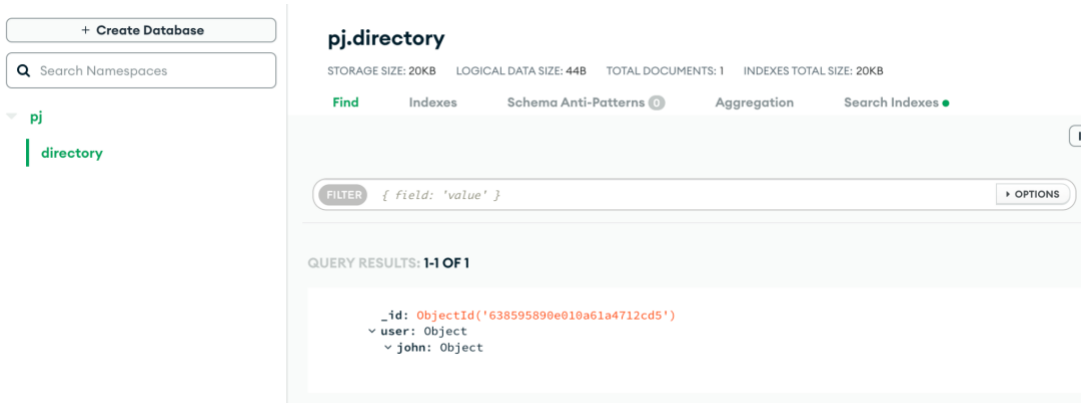
Task 1.2: Implement EDFS use MongoDB

As a 3 member team, we are required to implement 2 EDFS in different ways. We know that both Firebase & MongoDB are post-relational databases with JSON-like document data models and schemas.We have chosen to use firebase to present EDFS in task1, so here we are going to use MongoDB, which is also a document-oriented database, to implement the EDFS function.

Create a Directory: mkdir()

Different from the mkdir function implemented in Firebase, in MongoDB, we first split the directory we want to input, and decide to use different pymongo functions according to the number of levels in the directory. If the directory has only one level, we use the insert_one() Method. When the directory is larger than one level, we use the $set operator to replace the value then use update_one() function so that we can reach the last path name.

```
mkdir("/user")
mkdir("/user/john")
```



List all the files: ls()

In order to implement the ls function, we first split the directory we want to check, then we use the $exists operator and the pymongo function .find() to determine the items that meet the requirements, and print out the results.

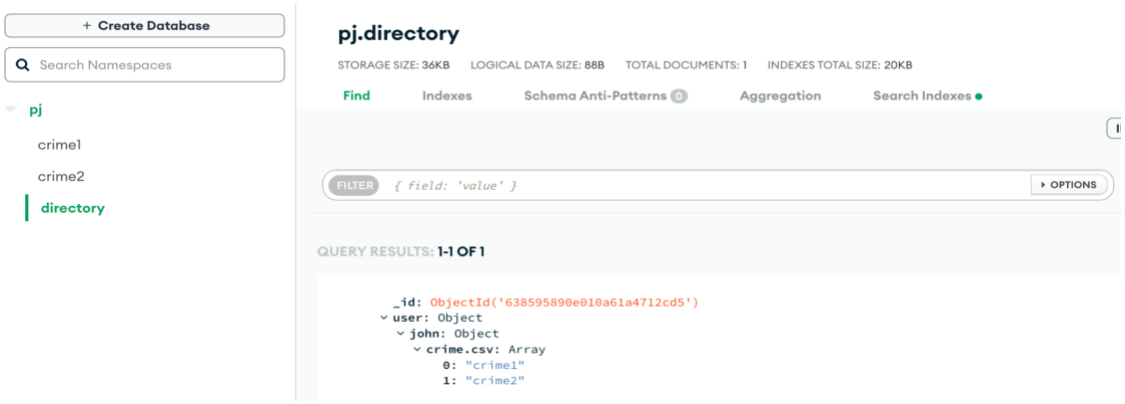
```
ls("/user")
```

```
[{'john': {}}]
```

Uploading a file to file system: put()

To implement the put function, we use the same logic as we did in firebase. But in mongoDB, we first need to understand how to put the file into the correct directory. Here we use the method of the above mkdir() function. Then we find the number of lines using Pandas. In order to achieve equal split, we set min_rows and max_rows through the number of rows whether they are divisible by (partition +1) or (partition +3), and return the floor division result. Then we use the random.randint function to generate random numbers in the range of min_rows and max_rows, stopping when the random number is less than or equal to 0. Then we read the csv file with the number of rows just generated, and as we did above save and write these dataframe to the csv file. Finally, we use insert_many() to insert multiple entries in a collection. In that case, we can achieve the command and upload a file to the file system with k partitions.

```
put("crime.csv","/user/john",2)
```



Return the locations of partitions of the file: getPartitionLocations()

To get the locations of partitions of the file, we classify them according to the length of the input paths. As we did in the ls function, we use the "$exists" operator and collection.find to confirm the existence of items with a root path (eg: /user), then use the .values() function returns a view object, and then Then print out the final locations of partitions

```
getPartitionLocations("/user/john/crime.csv")
```

```
['crime1', 'crime2']
```

Return the content of partition # of the specified file: readPartition()

In order to get the content of partition # of the specified file, we first get the filename we want to read through .split, then read the data directly, and print out the result.

```
readPartition("/user/john/crime.csv",2)
```

```
[{'_id': ObjectId('638595f20e010a61a4713127'),
  'id': 1106,
  'State': 'NEBRASKA',
  'County': 'Franklin',
  'Violent_Crime': 0.0,
  'Murder': 0.0,
  'Rape': 0.0,
  'Robbery': 0.0,
  'Aggravated_Assault': 0.0,
  'Property_crime': 10.0,
  'Burglary': 2.0,
  'Larceny-theft': 6.0,
  'Motor_vehicle_theft': 2.0,
  'Arson': 0.0},
 {'_id': ObjectId('638595f20e010a61a4713128'),
  'id': 1107,
  'State': 'NEBRASKA',
  'County': 'Frontier',
  'Violent_Crime': 3.0,
  'Murder': 0.0,
```

Display content of a file: cat()

In order to get the content of a file, we first get the exact location of the desired file through the getPartitionLocations function, and then use the .find function to get the specific content of the file.

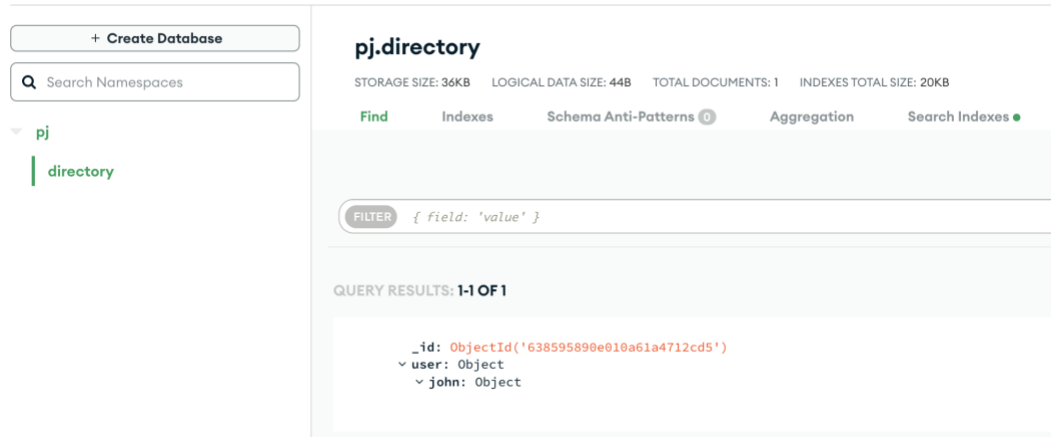
```
cat("/user/john/crime.csv")
```

```
Here are the file contents
{'_id': ObjectId('638595f20e010a61a4713127'), 'id': 1106, 'State': 'NEBRASKA', 'County': 'F
ranklin', 'Violent_Crime': 0.0, 'Murder': 0.0, 'Rape': 0.0, 'Robbery': 0.0, 'Aggravated_Ass
ault': 0.0, 'Property_crime': 10.0, 'Burglary': 2.0, 'Larceny-theft': 6.0, 'Motor_vehicle_t
heft': 2.0, 'Arson': 0.0}
{'_id': ObjectId('638595f20e010a61a4713128'), 'id': 1107, 'State': 'NEBRASKA', 'County': 'F
rontier', 'Violent_Crime': 3.0, 'Murder': 0.0, 'Rape': 1.0, 'Robbery': 0.0, 'Aggravated_Ass
ault': 2.0, 'Property_crime': 24.0, 'Burglary': 3.0, 'Larceny-theft': 20.0, 'Motor_vehicle_
theft': 1.0, 'Arson': 0.0}
{'_id': ObjectId('638595f20e010a61a4713129'), 'id': 1108, 'State': 'NEBRASKA', 'County': 'F
urnas', 'Violent_Crime': 7.0, 'Murder': 0.0, 'Rape': 2.0, 'Robbery': 0.0, 'Aggravated_Assau
lt': 5.0, 'Property_crime': 57.0, 'Burglary': 13.0, 'Larceny-theft': 35.0, 'Motor_vehicle_t
heft': 9.0, 'Arson': 0.0}
{'_id': ObjectId('638595f20e010a61a471312a'), 'id': 1109, 'State': 'NEBRASKA', 'County': 'G
age', 'Violent_Crime': 11.0, 'Murder': 0.0, 'Rape': 0.0, 'Robbery': 1.0, 'Aggravated_Assaul
t': 10.0, 'Property_crime': 77.0, 'Burglary': 16.0, 'Larceny-theft': 54.0, 'Motor_vehicle_t
heft': 7.0, 'Arson': 0.0}
```

Remove a file from the file system: rm()

In order to remove a file from the file system, we first use the getPartitionLocations function to get the specific location of the file we want to delete, and then use the .drop() function to remove the collection from the database. But only the files are deleted, and the previously created directory is preserved.

```
rm("/user/john/crime.csv")
```



Task 2: Implementing PMR on data stored on EDFS

After Task1, we have completed the implementation of EDFS in two ways (Firebase and MongoDB). The required data has also been stored in EDFS. Now we need to complete the

implementation of the partition-based map and reduce (PMR) function to make the further analytics easier.
The mapPartition(p) function mainly uses the readPartition() function in Task1, and also adds a judgment on the read partition, and finally returns a dataframe containing partition p.

Analytic function: find_county_in_state
Select county
From crime
Where state = ‘Arizona’

In this case, mapPartition(p) may take crime in partition p, output the county with different states (California, Utah, etc.)
Reduce function then identifies and returns the specific county name that fits the state name.

```
find_county_in_state("Arizona")
```

```
['Coconino',
 'Mohave',
 'Pinal',
 'Yavapai',
 'Yuma',
 'Gila',
 'Graham',
 'La Paz',
 'Navajo',
 'Santa Cruz',
 'Cochise']
```

Analytic function: find_county_by_GDP_rank_in_state
Select county
From GDP
Where state = ‘Arizona’ and rank = ‘2’

In this case, mapPartition(p) may take GDP in partition p, output the specific county with different state(California, Utah, etc.) and rank place.
Reduce function then identifies the specific state and the GDP rank and return the corresponding county name.

```
find_county_by_GDP_rank_in_state("Arizona",2)
```

```
['Pima']
```

Analytic function: find_county_by_income_rank_in_state
Select county
From income

Where state = ‘Arizona’ and rank = ‘2’

In this case, mapPartition(p) may take income in partition p, output the specific county with different state(California, Utah, etc.) and rank place.
Reduce function then identifies the specific state and the income ranking and returns the specific county according to the input ranking order of personal income in its state.

```
find_county_by_income_rank_in_state("Arizona",2)
```

```
['Coconino']
```

Analytic function: find_crimes_of_county_in_state
Select crimes
From crime
Where county = ‘Coconino’ and state = ‘Arizona’

In this case, mapPartition(p) may take crime in partition p, output the all case of crime data county with different state(California, Utah, etc.) and specific county.
Reduce function then identifies the state and county name then searches the crime record for this county in that state.

```
find_crimes_of_county_in_state("Coconino","Arizona")
```

	Aggravated_Assault	Arson	Burglary	County	Larceny-theft	Motor_vehicle_theft	Murder	Property_crime	Ra
0	103	6	59	Coconino	180	25	6	264	

Analytic function: Violent_Crime_by_state
Select Violent_crime
From crime
Where max/min/mean violent_crime Group by State

In this case, mapPartition(p) may take crime in partition p, output the corresponding crime value.
Reduce function then identifies the max/min/mean amount of violent crime of the county in the state.

```
Violent_Crime_by_state("max")
```

	0
ARIZONA	197.0
ARKANSAS	428.0
CALIFORNIA	5564.0
COLORADO	513.0
DELAWARE	912.0
FLORIDA	5353.0

Analytic function: Property_Crime_by_state
Select Property_crime
From crime
Where max/min/mean property_crime Group by State

In this case, mapPartition(p) may take crime in partition p, output the corresponding crime value.
Reduce function then identifies the max/min/mean amount of property crime of the county in the state.

```
Property_crime_by_state(method="max")
```

	0
ARIZONA	1983.0
ARKANSAS	1638.0
CALIFORNIA	15040.0
COLORADO	2845.0
DELAWARE	4125.0
FLORIDA	31822.0
GEORGIA	7364.0
HAWAII	4984.0
IDAHO	707.0
ILLINOIS	1059.0
INDIANA	642.0
IOWA	822.0
KANSAS	958.0

Analytic function: GDP_by_state
Select real_GDP
From GDP
Where max/min/mean GDP Group by State

In this case, mapPartition(p) may take GDP in partition p, output the corresponding GDP value.
Reduce function then identifies the max/min/mean amount of GDP of the county in the state.

```
GDP_by_state(method="min")
```

	0
Alabama	176482.0
Alaska	28935.0
Arizona	682877.0
Arkansas	91962.0
California	87671.0
Colorado	33225.0
Connecticut	4550511.0
Delaware	7681601.0
Florida	150680.0
Georgia	30275.0
Hawaii	3711519.0
Idaho	46172.0

Analytic function: income_by_state
Select Per_capita_personal_income
From income
Where max/min/mean income Group by State

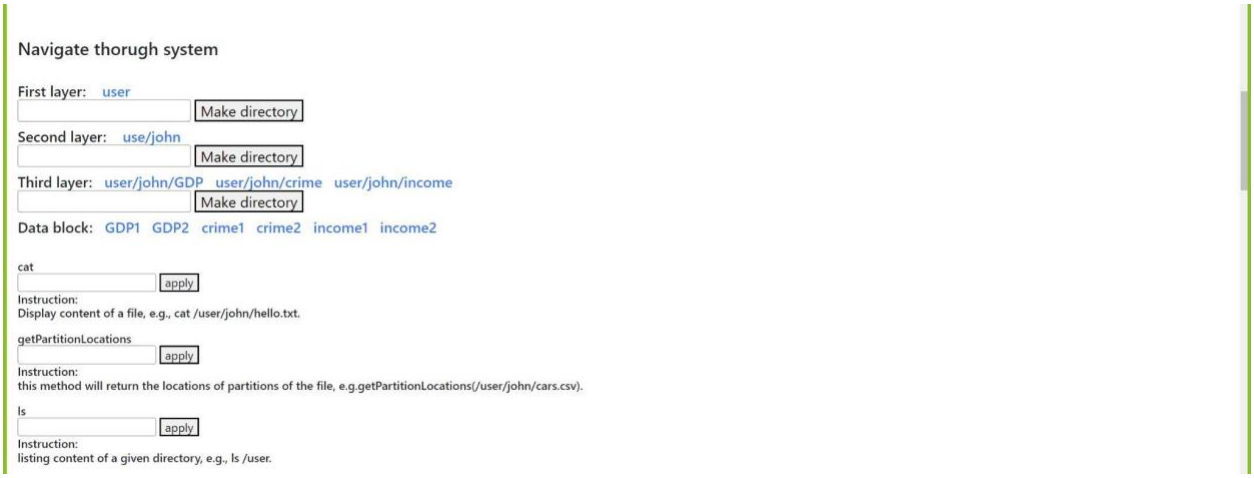
In this case, mapPartition(p) may take income in partition p, output the corresponding income value.
Reduce function then identifies the max/min/mean amount of income of the county in the state.

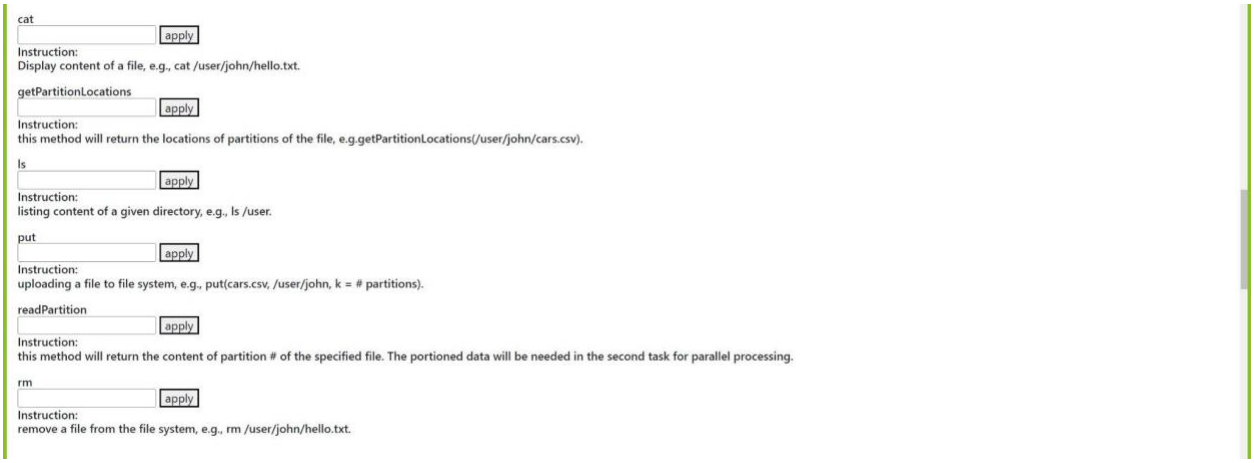
```
income_by_state(method="mean")
```

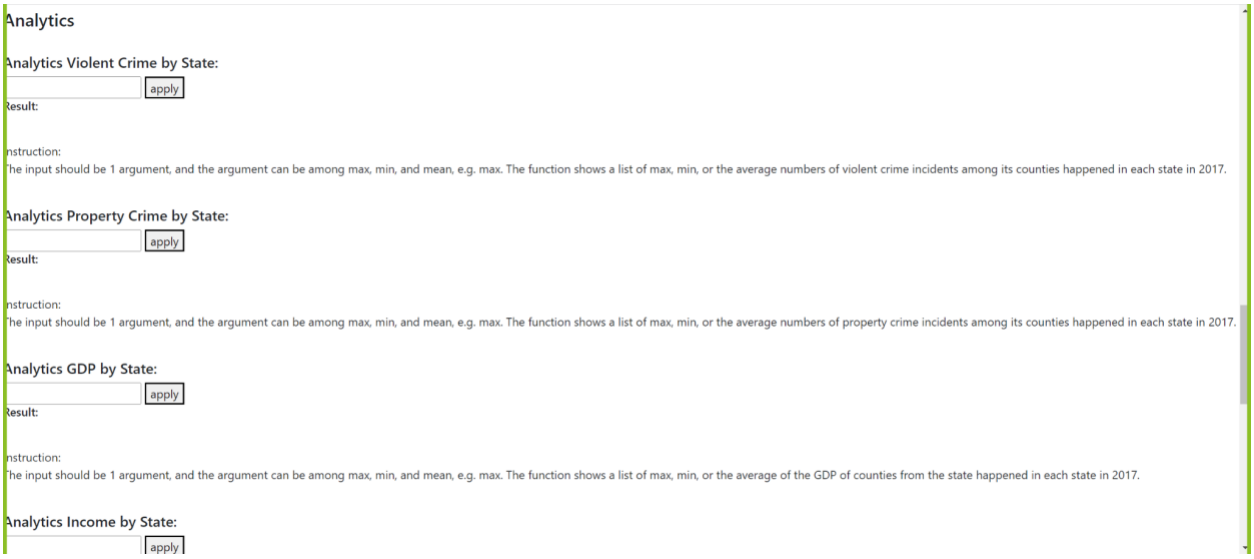
	0
Alabama	38354.761194
Alaska	64817.464286
Arizona	38486.800000
Arkansas	36485.680000
California	56931.017241
Colorado	51758.031250
Connecticut	66936.375000
Delaware	51366.666667
Florida	45038.298507
Georgia	37921.572327
Hawaii	50845.750000
Idaho	43325.704545
Illinois	45259.794118

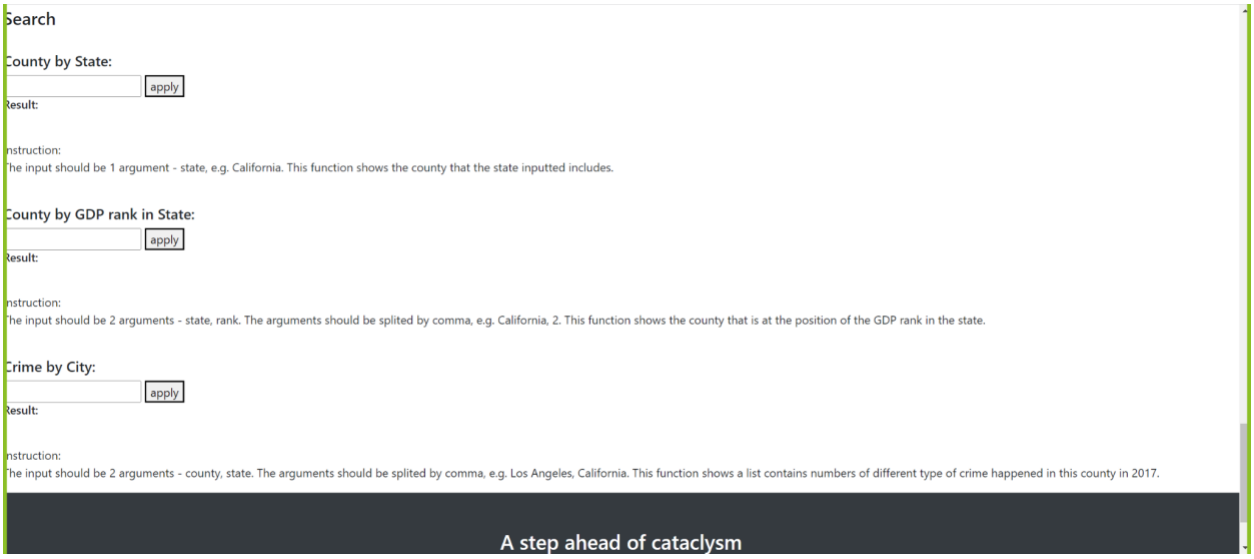
Task 3: Web browser-based APP

This is our team's attempt at a Web browser-based task. In this part, our main task is how to implement the EDFS system we built in firebase and mongoDB in task1 on the web page. Complete the navigate function of the system. People who use web pages do not need to understand any code, so the readability of the website is also a very important part.









In our web browser, although users can decide how to implement these functions depending on their need, we suggest users follow our instructions.

1. Use our analytics function to find the expectation state. Since our analytics function can help users get average, max, min of Crime number, income and GDP, it supports identifying the state that satisfied their expectation.

2. Use find_county_in_state function search to look for all the counties in the specific state.

3. Find the expected county by searching income/GDP rate rank

4. Through the function of find_crimes_of_county_in_state, users can finally get the exact crime data of the county and check the if county they choose is safe or not.

Learning Experience

Different from other groups, as three students without CS background, when we got this project, we were all a little confused and didn't know how to start. Our understanding of the database starts from the beginning of this class. Firebase realtime database is the first type of database we have come into contact with. Firebase uses the JSON format to store data. After the introduction in the lecture, we understand that JSON is a very popular cross-platform data exchange format, which has the advantages of being easy to read and modify. Compared with XML, another data interchange format commonly used in the Internet age, the JSON format is simpler, more lightweight and easier to use. After the practice of homework1, everyone has a preliminary understanding of Firebase. So we choose to use Firebase to complete the first emulated distributed file system. The json package in Python has given us a lot of help. These commands usually seem to be very common and easy to use, but in the process of writing functions by yourself, it is not as simple as imagined. For example, in my opinion, the most difficult thing to complete is the writing of the put function. Because in this process we need to create new content and change the data type, at the same time we also need to pay attention to k = # partitions. Because according to the functions that this function should have, we need to figure out the number of rows in the input data set, and then perform equally split on the data into k partitions.

Our second EDFS is done using MongoDB. We know that both Firebase & MongoDB are post-relational databases with JSON-like document data models and schemas. Generally speaking, the experience of using MongoDB is similar to using Firebase, and you can share the same set of writing ideas. Because they are all No SQL databases. It's just that we use the json package when writing firebase functions, and the pymongo package when writing MongoDB. For example, different from the mkdir function implemented in Firebase, in firebase we just use json.dump and json.load then use requests.put just finish the work. In MongoDB, we first split the directory we want to input, and decide to use different pymongo functions according to the number of levels in the directory. If the directory has only one level, we use the insert_one() Method. When the

directory is larger than one level, we use the $set operator to replace the value then use update_one() function so that we can reach the last path name.

The challenge we encountered in task3 is quite tough in implementing the UI. All three members of our group just have backgrounds mainly in mathematics and statistics, and have no systematic study of the front end of the web. In order to build the website framework, we need to study languages such as JavaScript, HTML5, CSS from 0. Whatsmore it's also a big challenge to make functions realize interactive functions on web pages.

Links

Google drive link:
https://drive.google.com/drive/folders/1uXZlI39T40oHJbFRdBkM1pDLbnKzXJO_

Demo video link:
https://youtu.be/T2-GD-qlh7g