Cost Distribution API

A solution by Pavlos Sakoglou, M.Sc.

Outline

Problem Statement

- We want to distribute cost to subsidiaries given an invoice
- We need an automated system to do the following:
  - Get an invoice as input
  - Extract information from the invoice
  - Determine the subsidiaries this invoice refers to
  - Distribute cost to these subsidiaries
  - Update cost database and ledger
  - Report SUCCESS or FAILURE
- We need a simple and intuitive Python API for the above system
- We can then use or integrate this Python API to other software components

How the Python API will look like?

```
from roivant.accounting import cost

def submit_new_invoice(invoice_path):
        # Create new Cost object to represent the invoice
        # Load the invoice and extract the information

        # Ask user to review the extracted information

        if reviewed:
                # Ask user to confirm

                if confirmed:
                         # Submit cost to database appropriately and return SUCCESS

        # If something went wrong, let the user know
        # Discard invoice data and return FAILURE


invoice = toPDF(LedgerAPI.GET_NEWEST())        # Ledger GET method to be converted as valid input

if not submit_new_invoice(invoice):            # Attempt to submit new invoice
                # Do something
```

Python API documentation

```
/roivant/accounting/cost.Cost.Cost()

       # Creates a new Roivant cost object


/roivant/accounting/cost.Cost.loadInvoice(invoice_path)

       # Takes a string path to the invoice or an invoice object as input, extracts information about the invoice,
       # and determines which subsidiaries are responsible for this expense proportionally. Allocates the cost appropriately


/roivant/accounting/cost.Cost.review()

       # Triggers a display screen with all the extracted information and the distributed cost for review
       # Returns true if confirmation is given, false otherwise


/roivant/accounting/cost.Cost.confirm()

       # Asks for second confirmation after seeing the review screen, and returns true or false as per the confirmation
```

Python API documentation

```
/roivant/accounting/cost.Cost.submit()

       # Updates the cost distribution database and saves the results.
       # Returns true if submission is successful, false otherwise


/roivant/accounting/cost.Cost.throw()

       # Displays a warning message with what went wrong.
       # Optional: Logs the error, timestamp, invoice information, and employ who attempted submission


/roivant/accounting/cost.Cost.reset()

       # Properly destroys the cost object
```

Additional API features for MIS

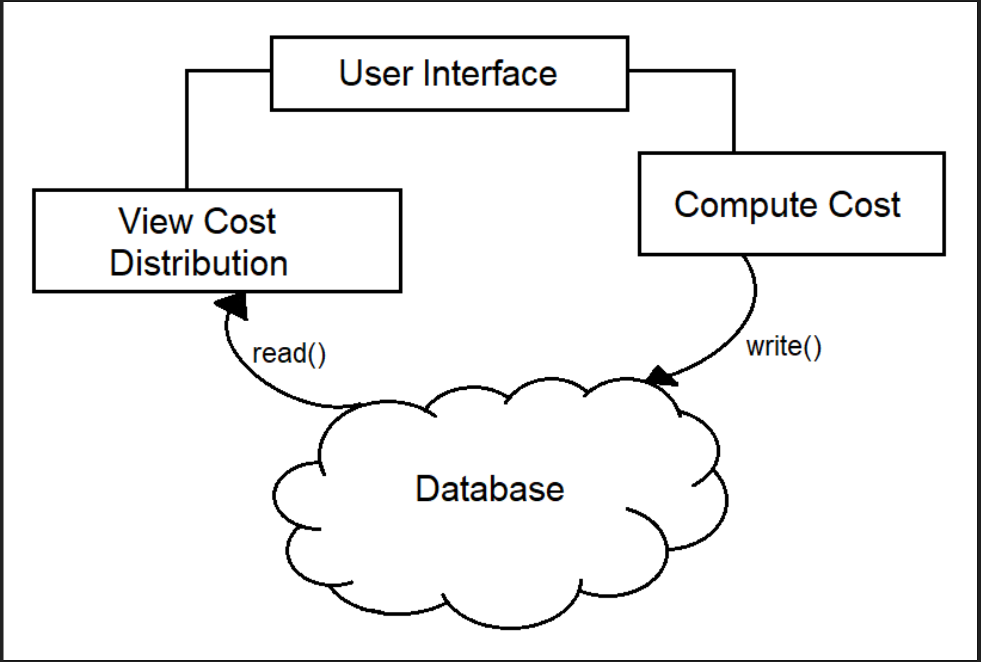
```
from roivant.accounting import view


stats = view.View()                         # instantiates a View object
stats.plotCostDistribution()                # display total cost distribution among all *vants
stats.plotCostTable()                       # display total cost table


costs = stats.getCostTable()                # get total cost table to use somewhere else



axovant = view.Axovant()                    # instantiate an Axovant view object
axovant.summary()                           # display the cost summary of Axovant
axovant.plotCostTable()                     # display total cost table of Axovant



datavant = view.Datavant()                  # instantiate a Datavant view object
datavant.printSummary()                     # connects to printer and prints total cost summary of Datavant
datavant.printCostTable()                   # connects to printer and prints total cost table of Datavant


# More functionality
```

Additional API features documentation

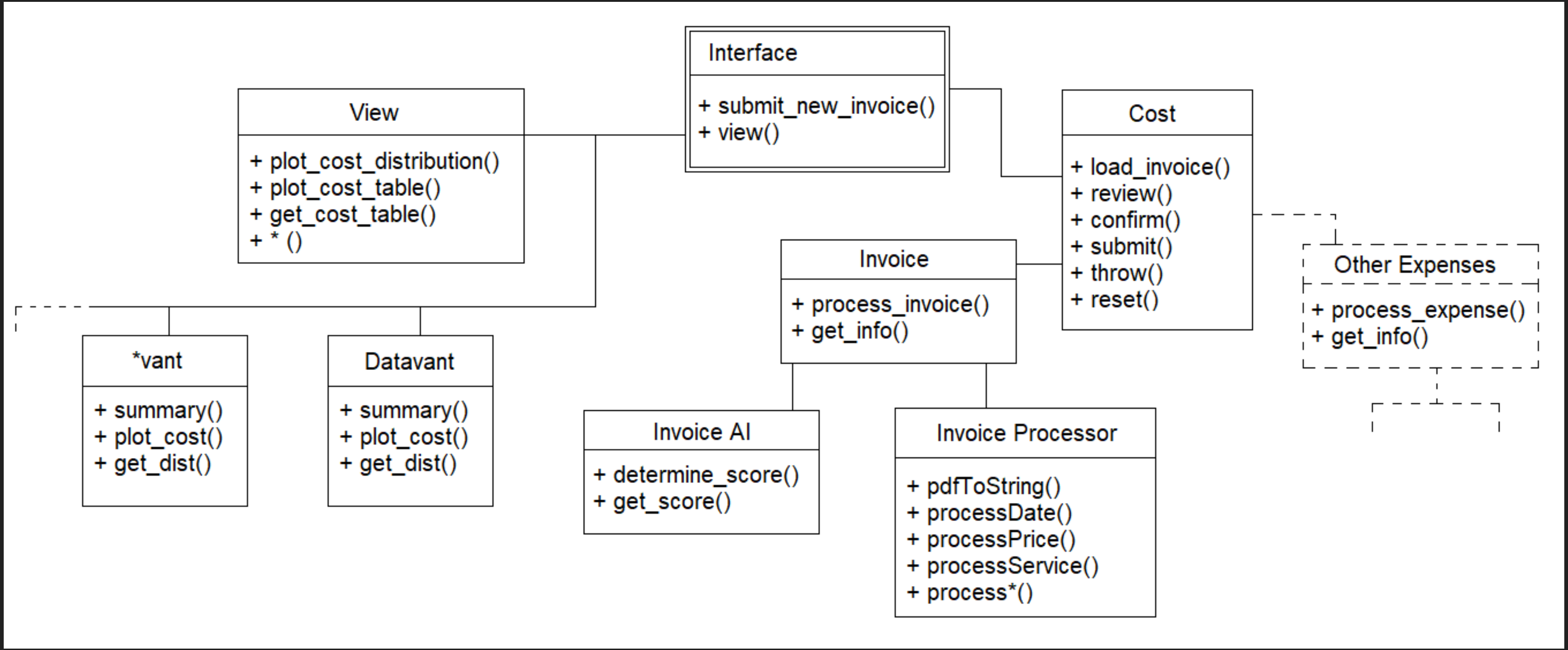
```
/roivant/accounting/view.View()

      # Creates a new MIS object

/roivant/accounting/view.View.plotCostDistribution()

      # Displays already computed statistics from data warehouse. Will plot in histograms and data pies historical cost and
      # proportional expenses per subsidiary. Optionally, more information can be visualized

/roivant/accounting/view.View.plotCostTable()

      # Displays and plots information from the general cost table of Roivant subsidiaries

/roivant/accounting/view.View.getCostTable()

      # Returns the NxM data table of general cost of Roivant subsidiaries

/roivant/accounting/view.*vant()

      # Creates a new MIS object for particular Roivant subsidiary, overriding MIS functionality + extra features

/roivant/accounting/view.*vant.summary()

      # Displays a table with historical transactions and brief cost summary
```

Outline

System design



System design



System design

- Source code structure

```
/roivant (root dir)
              /accounting (child dir)
                            /cost.py
                                  -> Cost()
                                          -> loadInvoice()
                                          -> review()
                            /invoice.py
                                  -> Invoice()
                                          -> process_invoice()
                                          -> get_info()
                            /invoiceProcessor.py
                                  -> pdfToString()
                                  -> processDate()
                                  -> …
                            /view.py
                                  -> View()
                                  -> *vant()
                            /…
```

Scalability / Maintenance

- Minimum dependencies
  - Will work with different ledger software
  - Basic OS requirements and Python packages
- Will support and work fast with a high volume of data
  - Choose a scalable data warehouse system
- Easily extended to support more subsidiaries and their children
- Easy to add new functionality or customize existing one
- Intuitive API methods
  - Easy to use
  - Easy to learn

Pitfalls

- Need to restructure invoices and other expenses into templated formats
- Requires additional data warehouse system for fast data querying
- More work to convert this into software suite i.e. C#, C++, etc.
- Requires advanced AI and text recognition functions
- Won’t work with any input invoice/cost doc format
- Existing Software Integration challenges

Outline

- Domain Background
- System Design
- **Implementation**
- Intelligent features
- Demo
- Q&A

Implementation

- Cost class (part 1)

```
# General Cost class
class Cost:

    def __init__(self):                                          # Constructor
              self.invoice_info = []                             # Encapsulates the invoice extracted data


    def loadInvoice(self, invoice_path):                         # Loads and processes the input invoice
        inv = invoice.Invoice()                                  # Look at the Invoice class for the methods below
        inv.processInvoice(invoice_path)
        self.invoice_info = inv.getInfo()
        print("Invoice loaded")

    def review(self):                                            # Confirmation screen
        print("Please review the following:")                    # Displays the information again and asks user
        for i in self.invoice_info:
            print(i)
        print("Press 1 when ready, 0 to cancel.")
        return int(input())
```

Implementation

- Cost class (part 2)

```
def submit(self):
        addresses = self.invoice_info[-1]
        t = time.time()
        datestamp = str(datetime.datetime.fromtimestamp(t).strftime('%m-%d-%Y %H:%M:%S'))
        price = self.invoice_info[0][1]
        service = self.invoice_info[1][1]
        date = self.invoice_info[2][1]
        score = self.invoice_info[3][1]
        conn = sqlite3.connect('RoivantDB.db')
        c = conn.cursor()
        for i in addresses[1]:
            if i == 'Roivant':
                c.execute("INSERT INTO Roivant (datestamp, service, price, date, score) VALUES(?,?,?,?,?)",
                          (datestamp, service, price, date, score))
                conn.commit()
            if i == 'Datavant':
                c.execute("INSERT INTO Datavant (datestamp, service, price, date, score) VALUES(?,?,?,?,?)",
                          (datestamp, service, price, date, score))
                conn.commit()
        c.close()
        conn.close()
        print("Success!")
```

Implementation

- Invoice class (part 1)

```
# Invoice class to objectify a new invoice
class Invoice:

    def __init__(self):                                  # Encapsulates all the invoice data of interest
        self.price = 0.0                                 # that are meant to be copied in the database
        self.service = ""
        self.date = ""
        self.score = 0.0
        self.addresses = []

    def processInvoice(self, pdf_path):                  # Function to process the invoice and assign
        text = ip.pdfToString(pdf_path)                  # the extracted information to each invoice attribute
        self.date = ip.processDate(text)
        self.price = ip.processPrice(text)
        self.service = ip.processService(text)
        analysis = ai.InvoiceAI()
        analysis.determineScore(text)
        self.score = analysis.getScore()
        self.addresses = ip.getAddresses(text)
```

Implementation

- Invoice class (part 2)

```
# Function to return a list of the initialized
# invoice attributes

def getInfo(self):
    return [ ["price", self.price],
             ["service", self.service],
             ["date", self.date],
             ["score", self.score],
             ["addresses", self.addresses] ]
```

Implementation

- Invoice Processor file

```
def pdfToString(pdf_path):                                # Free function that reads a pdf and converts its contents
    pdfFileObj = open(pdf_path, 'rb')                     # to a string of characters
    pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
    pageObj = pdfReader.getPage(0)
    return pageObj.extractText()


def processService(text):                                 # Free function that reads the string version of the invoice
    start = text.find("Service(s)") + 13                  # and extracts the service from its location in the string
    end = text.find("Point(s)") - 3
    return text[start:end]


def getAddresses(text):                                   # Free function that reads the string version of the invoice
    ''' Extract Subsidiaries' names '''                   # and extracts the subsidiary companies that the invoice refers to
    var1 = 'Roivant'
    var2 = 'Datavant'
    addresses = [var1, var2]
    return addresses

# MORE FUNCTIONALITY ...
```

Outline

- Domain Background
- System Design
- Implementation
- **Intelligent features**
- Demo
- Q&A

Intelligent features

- Cost inference with an Invoice AI system
  - Create a NLP algorithm to “read” parts of the invoice i.e. service, rationale, etc.
  - Classify invoice to an *importance category*
  - Compute importance/price ratio for the invoice
  - Compute and assign importance score and return a brief summary report

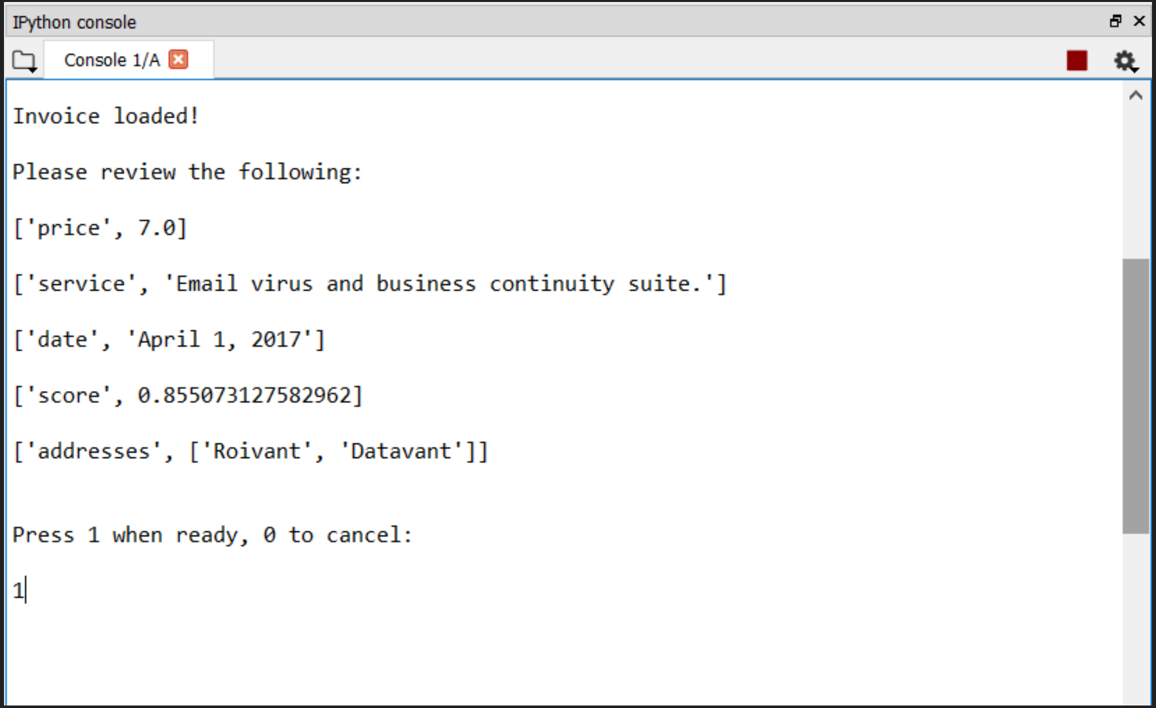
Intelligent features

- Data mining
  - Plot cost distributions for each subsidiary
  - Plot historical cost data against their importance scores
  - Cluster cost categories and *learn* hidden segmentations of the expenses of each *vant
  - Use this information to classify new invoices and costs better
  - Visualization of the decision making on cost of each department

Outline

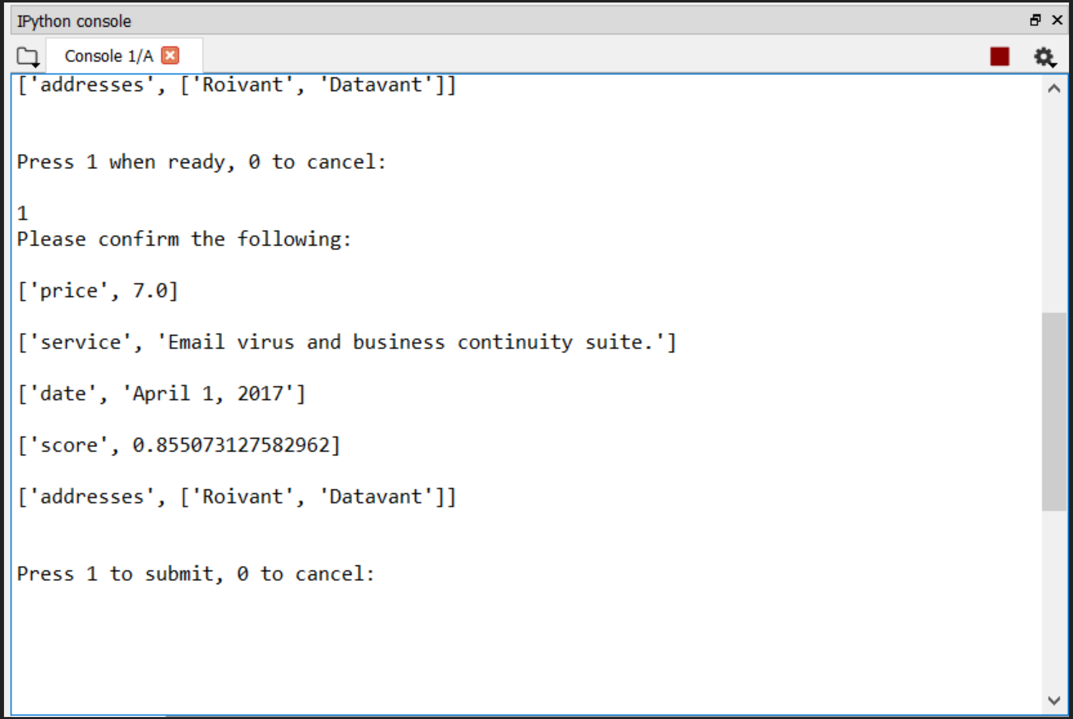
- Domain Background
- System Design
- Implementation
- Intelligent features
- **Demo**
- Q&A

Demo



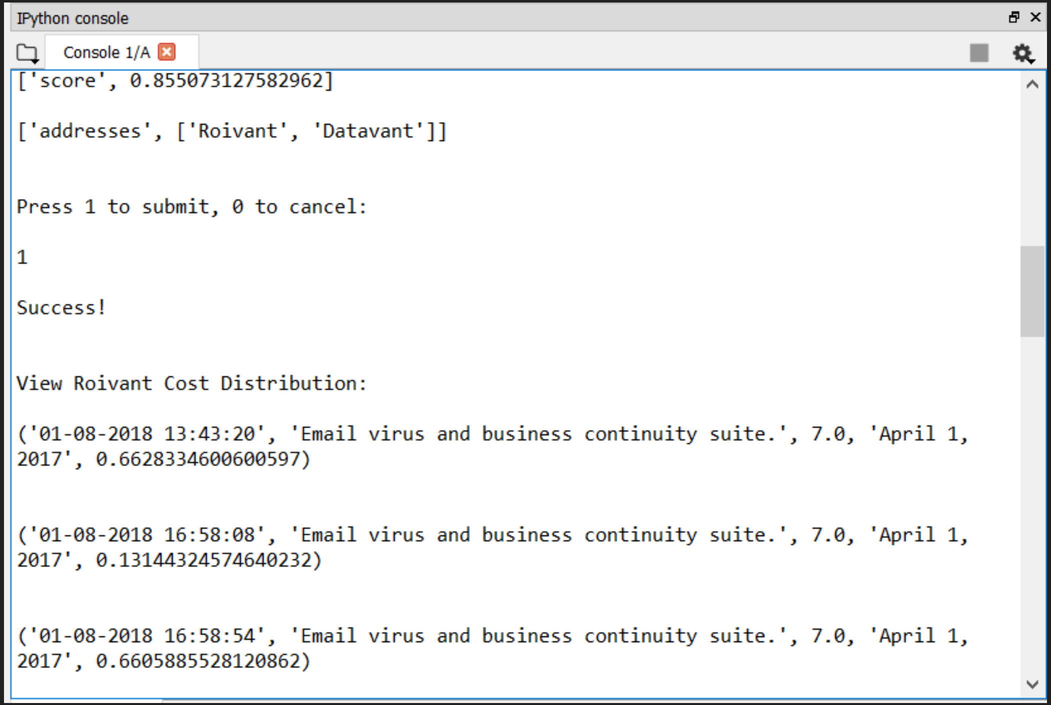
```
Invoice loaded!

Please review the following:

['price', 7.0]

['service', 'Email virus and business continuity suite.']

['date', 'April 1, 2017']

['score', 0.855073127582962]

['addresses', ['Roivant', 'Datavant']]


Press 1 when ready, 0 to cancel:

1
```

Demo



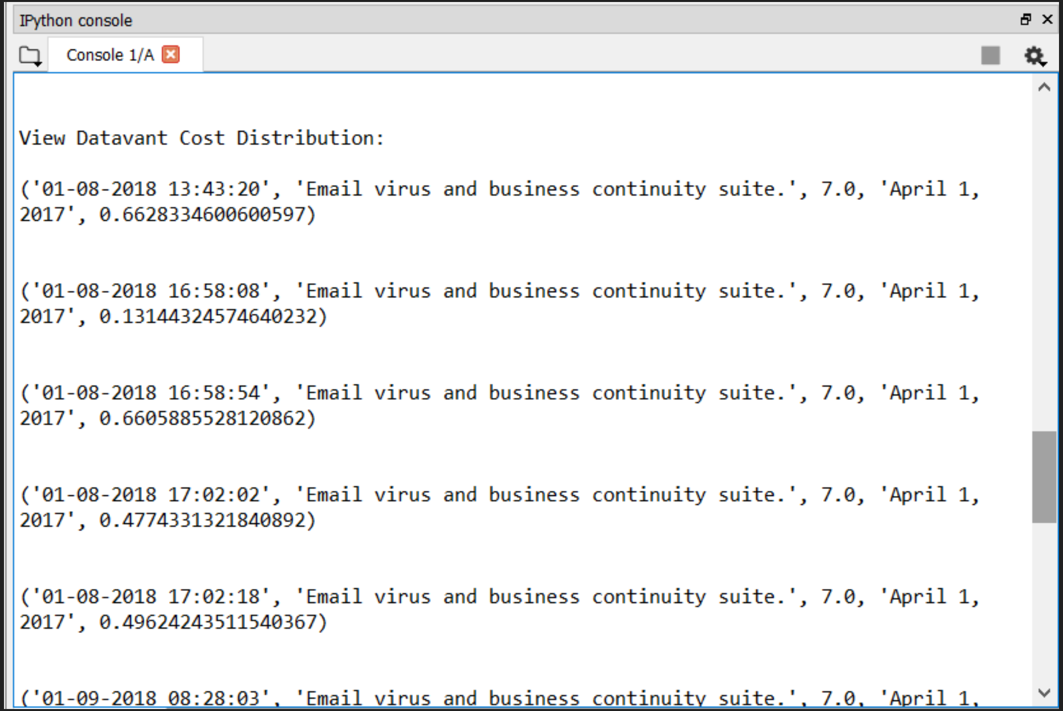
```
['addresses', ['Roivant', 'Datavant']]


Press 1 when ready, 0 to cancel:

1
Please confirm the following:

['price', 7.0]

['service', 'Email virus and business continuity suite.']

['date', 'April 1, 2017']

['score', 0.855073127582962]

['addresses', ['Roivant', 'Datavant']]


Press 1 to submit, 0 to cancel:
```

Demo



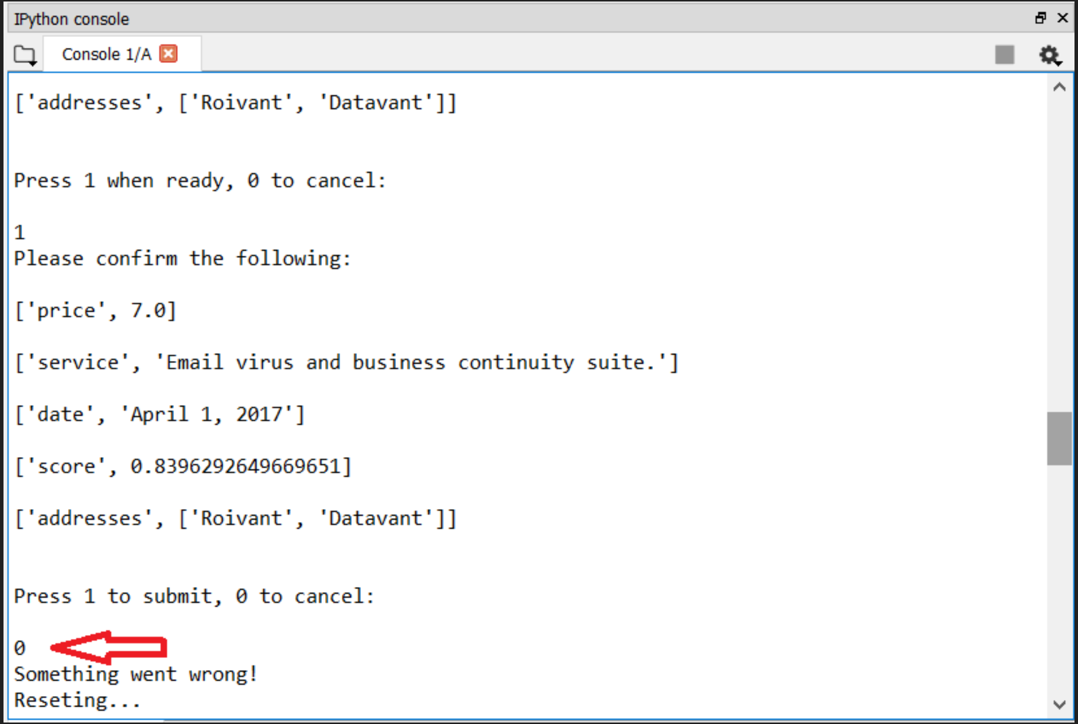
```
['score', 0.855073127582962]

['addresses', ['Roivant', 'Datavant']]


Press 1 to submit, 0 to cancel:

1

Success!


View Roivant Cost Distribution:

('01-08-2018 13:43:20', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.6628334600600597)


('01-08-2018 16:58:08', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.13144324574640232)


('01-08-2018 16:58:54', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.6605885528120862)
```

Demo



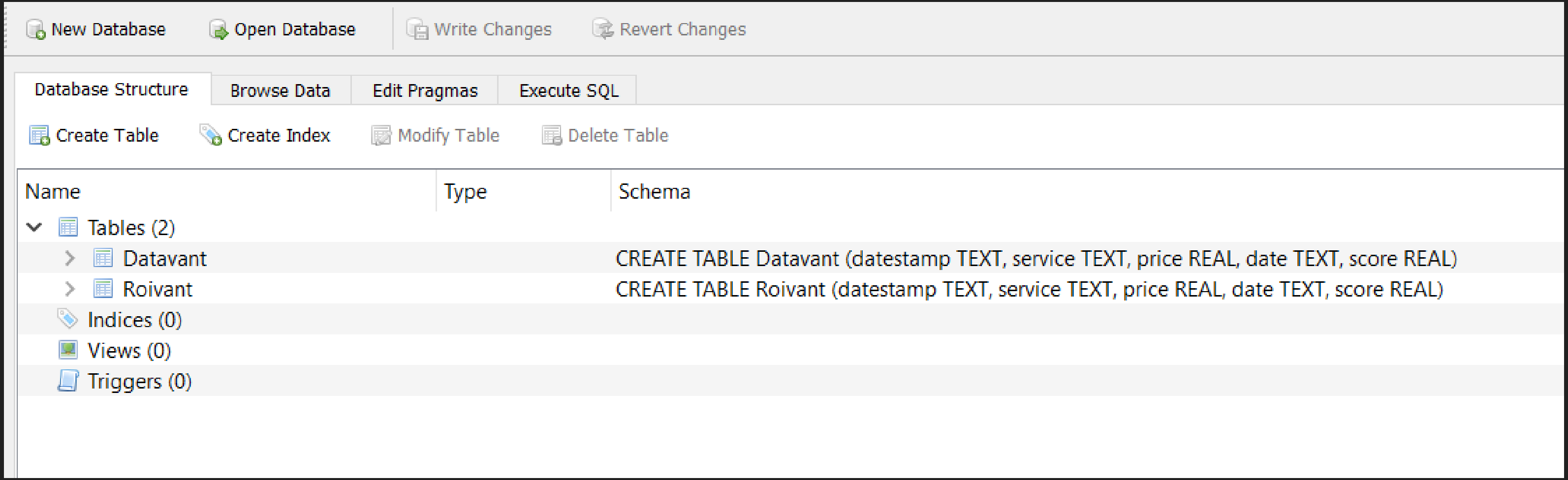
```
View Datavant Cost Distribution:

('01-08-2018 13:43:20', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.6628334600600597)


('01-08-2018 16:58:08', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.13144324574640232)


('01-08-2018 16:58:54', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.6605885528120862)


('01-08-2018 17:02:02', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.4774331321840892)


('01-08-2018 17:02:18', 'Email virus and business continuity suite.', 7.0, 'April 1,
2017', 0.49624243511540367)


('01-09-2018 08:28:03', 'Email virus and business continuity suite.', 7.0, 'April 1,
```

Demo



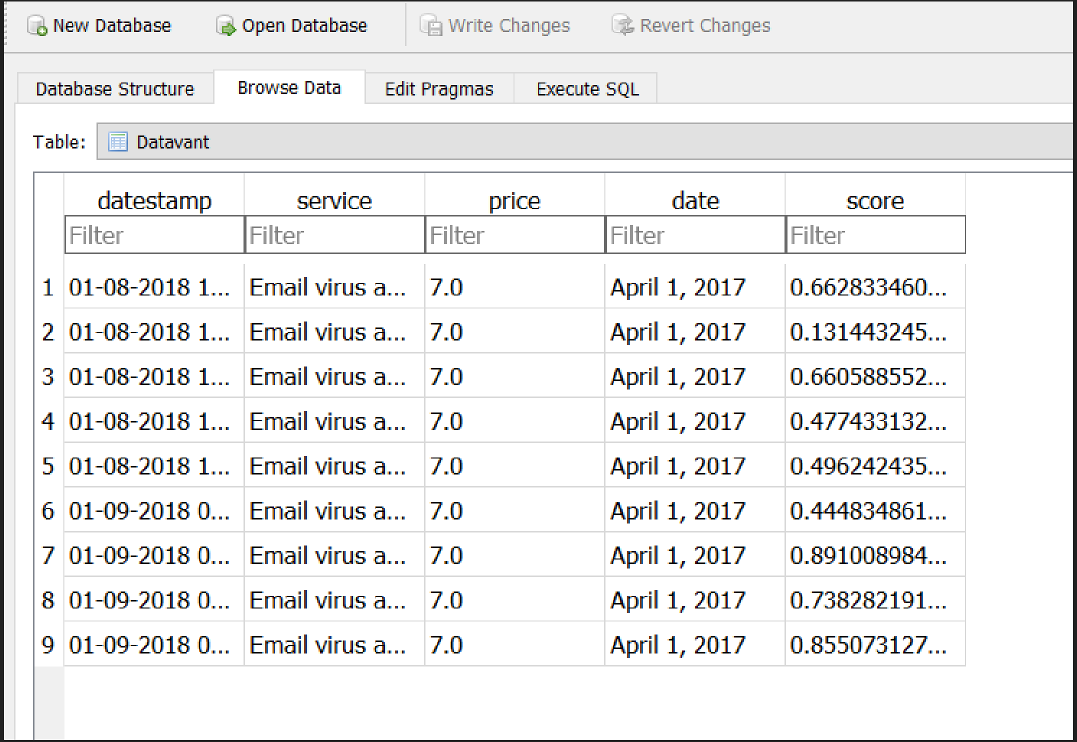
```
['addresses', ['Roivant', 'Datavant']]


Press 1 when ready, 0 to cancel:

1
Please confirm the following:

['price', 7.0]

['service', 'Email virus and business continuity suite.']

['date', 'April 1, 2017']

['score', 0.8396292649669651]

['addresses', ['Roivant', 'Datavant']]


Press 1 to submit, 0 to cancel:

0
Something went wrong!
Reseting...
```

Demo



Name	Type	Schema
Tables (2)		
Datavant		CREATE TABLE Datavant (datestamp TEXT, service TEXT, price REAL, date TEXT, score REAL)
Roivant		CREATE TABLE Roivant (datestamp TEXT, service TEXT, price REAL, date TEXT, score REAL)
Indices (0)		
Views (0)		
Triggers (0)		

Demo



New Database | Open Database | Write Changes | Revert Changes

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: Datavant

	datestamp	service	price	date	score
	Filter	Filter	Filter	Filter	Filter
1	01-08-2018 1...	Email virus a...	7.0	April 1, 2017	0.662833460...
2	01-08-2018 1...	Email virus a...	7.0	April 1, 2017	0.131443245...
3	01-08-2018 1...	Email virus a...	7.0	April 1, 2017	0.660588552...
4	01-08-2018 1...	Email virus a...	7.0	April 1, 2017	0.477433132...
5	01-08-2018 1...	Email virus a...	7.0	April 1, 2017	0.496242435...
6	01-09-2018 0...	Email virus a...	7.0	April 1, 2017	0.444834861...
7	01-09-2018 0...	Email virus a...	7.0	April 1, 2017	0.891008984...
8	01-09-2018 0...	Email virus a...	7.0	April 1, 2017	0.738282191...
9	01-09-2018 0...	Email virus a...	7.0	April 1, 2017	0.855073127...

That is all!

Thank you for your time!