122022010 Shrayank Mistry
Description:
1) Implement hash based natural join
2) Implement sort based natural join
Programming Language: Python

Hash based Natural Join
Input Relations A, B with hash function that returns ascii value

```python
from tabulate import tabulate

#* Hash function return ascii value
def hash_function(h):
    return ord(str(h))

table_A = [
    ['CustomerId', 'CustomerName'],
    [1, 'C1'],
    [2, 'C2'],
    [3, 'C3']
]

table_B = [
    ['InvoiceId', 'CustomerId', 'Total'],
    [1, 2, '$44.20'],
    [2, 2, '$13.37'],
    [3, 1, '$144.5'],
    [4, 3, '$501.1'],
    [5, 3, '$66.77'],
    [6, 1, '$100.0']
]
```

1) Build Phase

```python
#* Build Phase
build_table = {}

for i in range(1, len(table_A)):
    h = hash_function(table_A[i][0])
    if build_table.get(h) == None:
        build_table[h] = list()
    t = [table_A[i][j] for j in range(0, len(table_A[i]))]
    build_table[h].append(t)
```

2) Natural Join

```python
joined_table = list()
#* Natural Join Phase

for i in range(1, len(table_B)):
    h = hash_function(table_B[i][1])
    if build_table.get(h) == None:
        continue
    records_A = build_table[h]
    t = [table_B[i][j] for j in range(0, len(table_B[i]))]
    for record in records_A:
        joined_table.append(list(record + t))

# for record in joined_table:
#     print(record)

headers = table_A[0] + table_B[0]
print(tabulate(joined_table, headers))
```

3) Output

```
  CustomerId  CustomerName      InvoiceId    CustomerId  Total
------------  --------------  -----------  ------------  -------
           2  C2                        1             2  $44.20
           2  C2                        2             2  $13.37
           1  C1                        3             1  $144.5
           3  C3                        4             3  $501.1
           3  C3                        5             3  $66.77
           1  C1                        6             1  $100.0
```

Sort Based Natural Join
Input Relations A, B with sorting function on CustomerId

```python
from tabulate import tabulate

def sort_for_B(L):
    return L[1]

def sort_for_A(L):
    return L[0]

table_A = [
    ['CustomerId', 'CustomerName'],
    [1, 'C1'],
    [2, 'C2'],
    [3, 'C3']
]

table_B = [
    ['InvoiceId', 'CustomerId', 'Total'],
    [1, 2, '$44.20'],
    [2, 2, '$13.37'],
    [3, 1, '$144.5'],
    [4, 3, '$501.1'],
    [5, 3, '$66.77'],
    [6, 1, '$100.0']
]
```

1) Sorting Relations A, B on CustomerId

```python
headers = table_A[0] + table_B[0]

#* Sorting tables A, B on customerId
table_B = sorted(table_B[1:], key=sort_for_B)
table_A = sorted(table_A[1:], key=sort_for_A)
```

2) Natural Join

```python
joined_table = list()
#* Natural Join Phase
j = 0
for i in range(len(table_A)):
    while j < len(table_B):
        if table_A[i][0] == table_B[j][1]:
            joined_table.append(list(table_A[i] + table_B[j]))
            j = j + 1
        else:
            break

print(tabulate(joined_table, headers))
```

3) Output

```
 CustomerId  CustomerName      InvoiceId    CustomerId  Total
------------  --------------  -----------  ------------  -------
          1  C1                        3             1  $144.5
          1  C1                        6             1  $100.0
          2  C2                        1             2  $44.20
          2  C2                        2             2  $13.37
          3  C3                        4             3  $501.1
          3  C3                        5             3  $66.77
```