Group No	Name	Student Email id	% Contribution
144	Abhinav Chandravanshi	2020FC04860	Full (100%)
144	Anurag	2020FA04028	Full (100%)

Databases used for comparison to do performance evaluation:

- **Mongo DB**
  MongoDB Compass 1.29.5
  MongoDB 5.0.5 Enterprise (64 bit)
  Mongo DB Command Line Shell 5.0.5

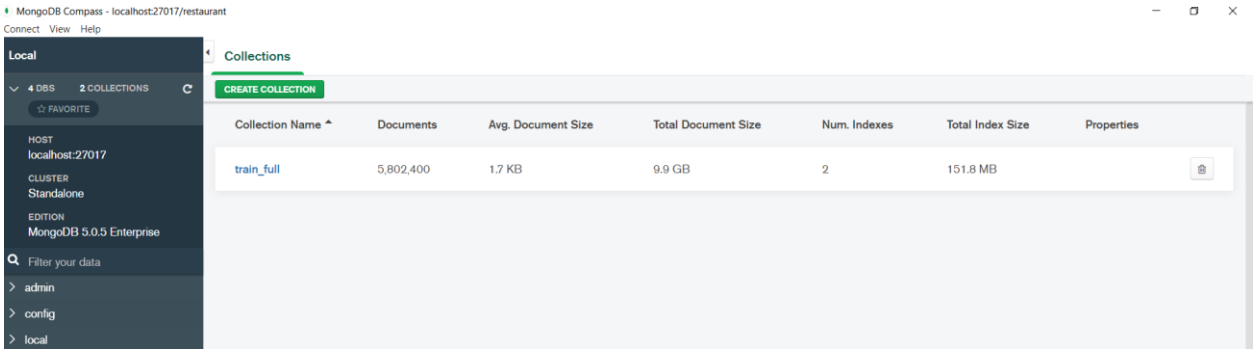
- **PostgreSQL**
  PGAdmin 6.1
  PostgreSQL DB 14
  PSQL Command Line Shell 14.1

Hardware and OS:

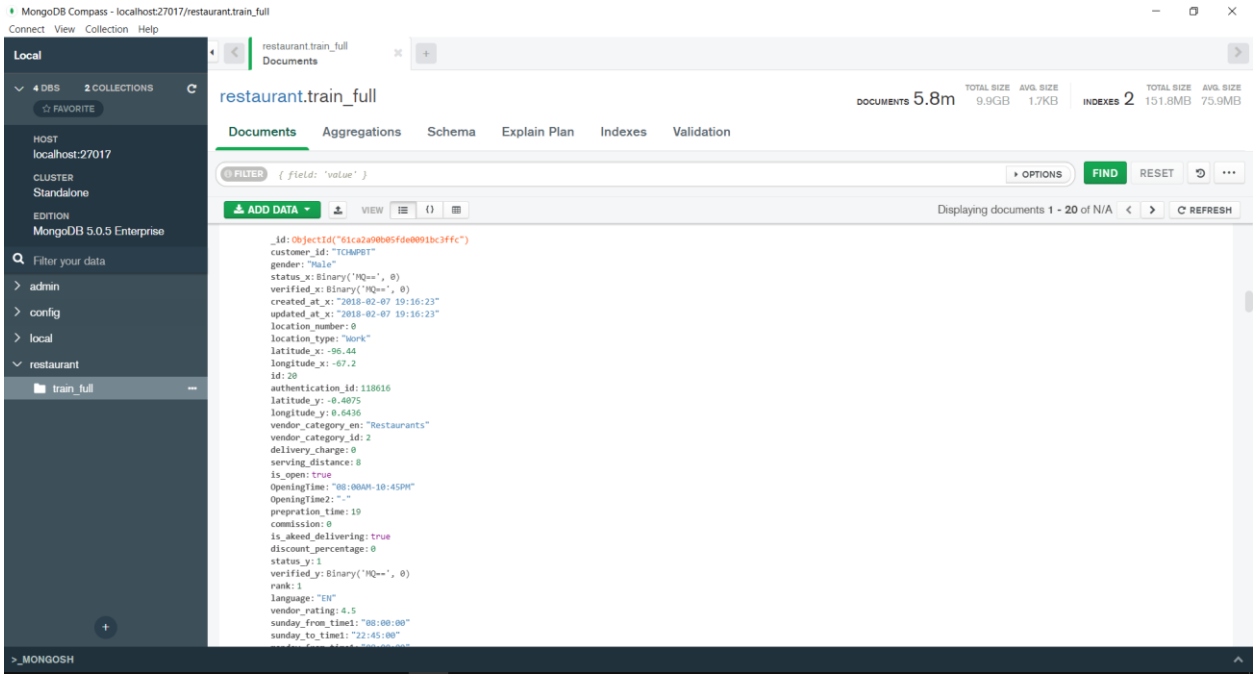
Processor: Intel Core i-3 7100 CPU @ 2.40 GHz
RAM: 6 GB
Storage Type: SSD
Operating System: Windows 10 Home
Architecture Type: 64-bit operating system, x64-based processor

Compass UI

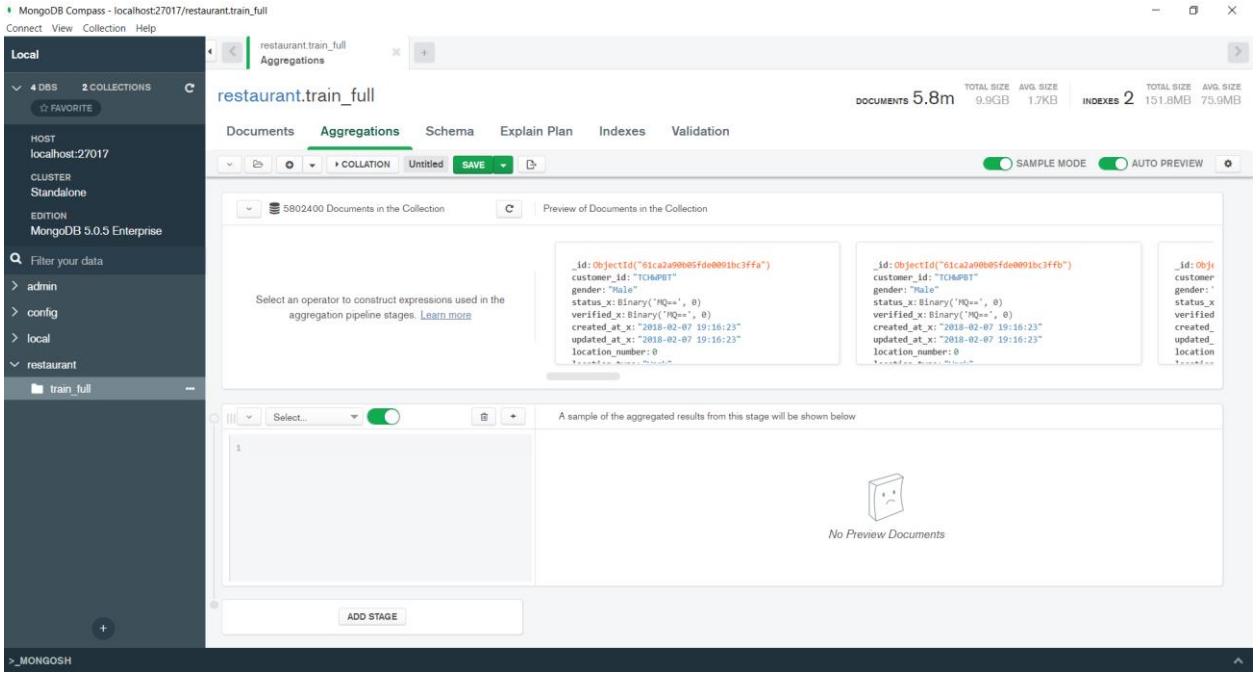
DB Size and Index Size



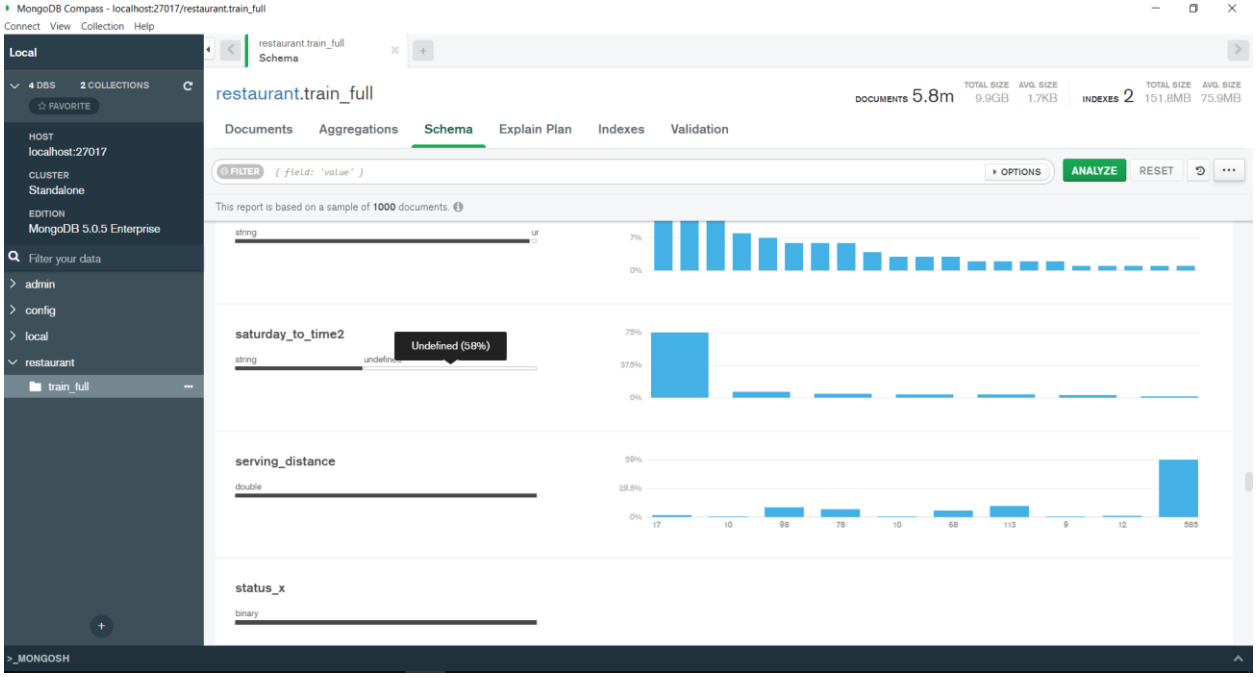
Sample documents



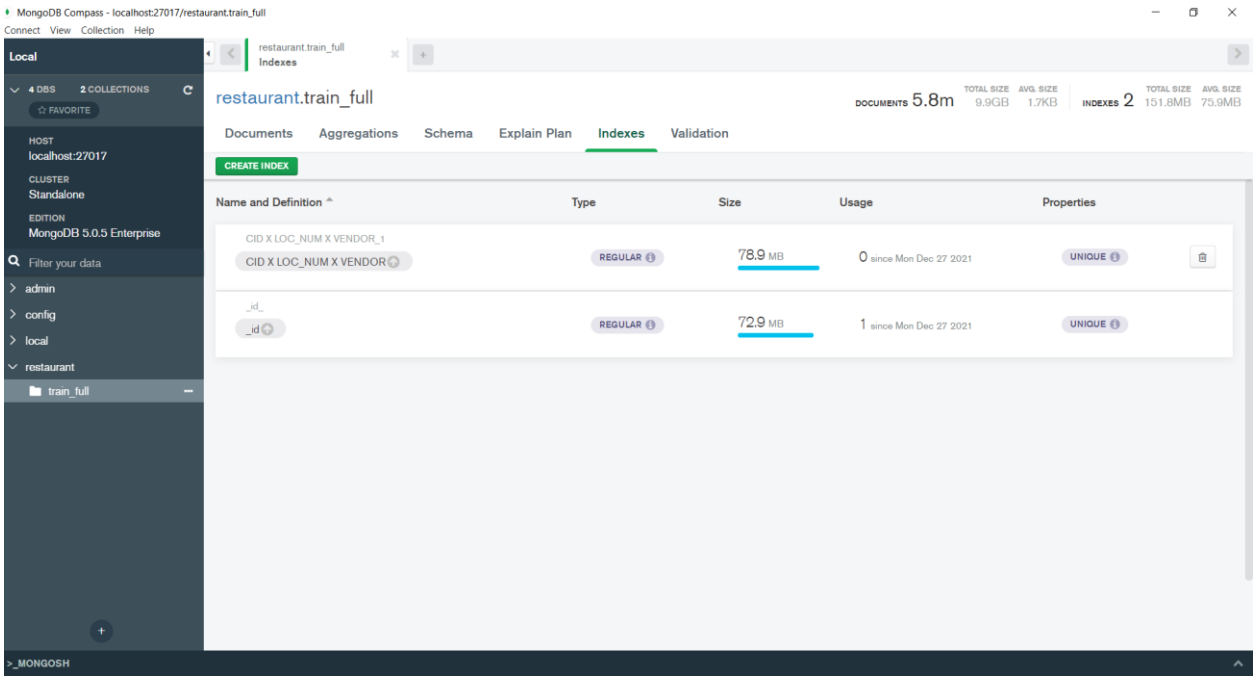
Aggregations



Schema Analysis



Index



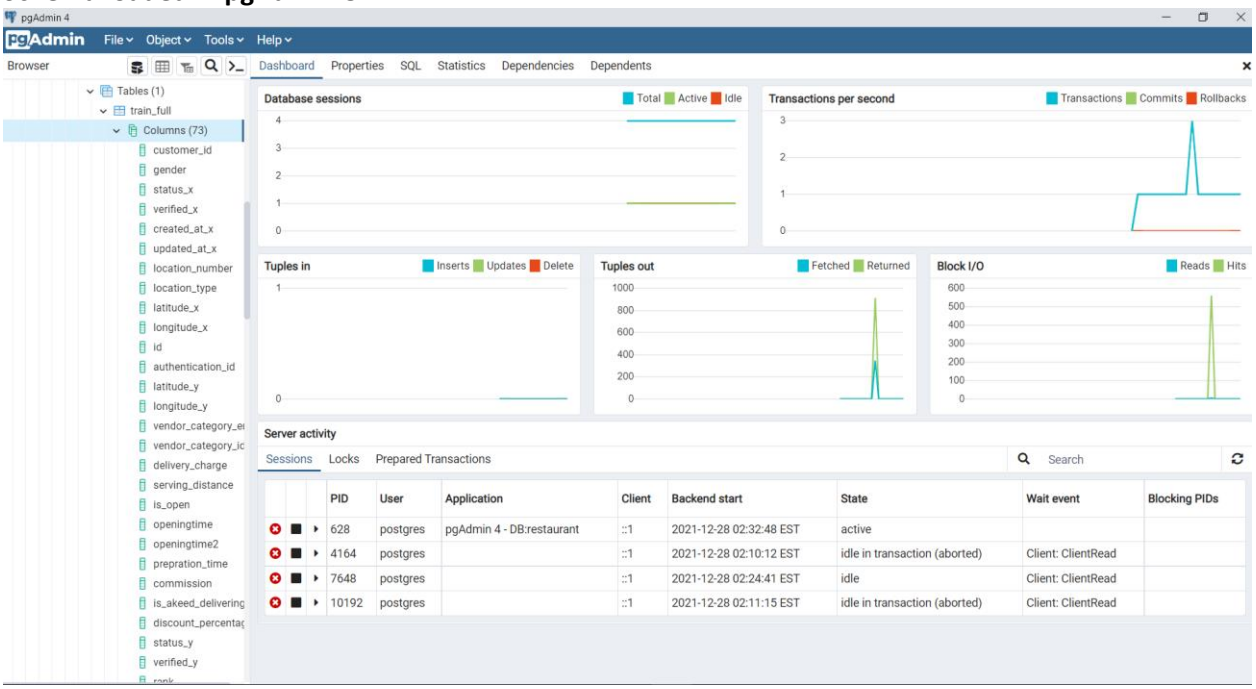
- **Mongo DB Document Sample Schema**


```
{
 "created_at_y": "2018-01-30 14:42:04",
 "created_at_x": "2018-02-07 19:16:23",
 "gender": "Male",
 "saturday_from_time2": "10:00:00",
 "saturday_from_time1": "00:00:00",
 "_id": {
  "$oid": "61ca2a90b05fde0091bc3ffa"
 },
 "wednesday_from_time2": "08:00:00",
 "wednesday_from_time1": "00:00:00",
 "OpeningTime": "11:00AM-11:30PM",
 "friday_to_time2": "23:59:00",
 "friday_to_time1": "00:30:00",
 "target": 0,
 "sunday_from_time2": "08:00:00",
 "sunday_from_time1": "00:00:00",
 "city_id": 1,
 "delivery_charge": 0,
 "monday_to_time2": "23:59:00",
 "monday_to_time1": "00:30:00",
 "authentication_id": 118597,
 "primary_tags": "{\"primary_tags\":\"4\"}",
 "CID X LOC_NUM X VENDOR": "TCHWPBT X 0 X 4",
 "discount_percentage": 0,
 "thursday_to_time2": "23:59:00",
 "thursday_to_time1": "00:30:00",
 "display_orders": 1,
 "vendor_rating": 4.4,
```

```
"country_id": 1,
"saturday_to_time2": "23:59:00",
"saturday_to_time1": "00:30:00",
"verified_y": {
 "$binary": "MQ==",
 "$type": "0"
},
"verified_x": {
 "$binary": "MQ==",
 "$type": "0"
},
"commission": 0,
"wednesday_to_time2": "23:59:00",
"wednesday_to_time1": "00:30:00",
"is_akeed_delivering": true,
"serving_distance": 6,
"id_obj": 4,
"vendor_tag": "2,4,5,8,91,22,12,24,16,23",
"latitude_y": -0.5884,
"latitude_x": -96.44,
"monday_from_time2": "08:00:00",
"monday_from_time1": "00:00:00",
"open_close_flags": 1,
"device_type": 3,
"location_type": "Work",
"location_number_obj": 0,
"updated_at_y": "2020-04-07 15:12:43",
"updated_at_x": "2018-02-07 19:16:23",
"id": 4,
"is_open": true,
"thursday_from_time2": "08:00:00",
"thursday_from_time1": "00:00:00",
"vendor_category_id": 2,
"vendor_category_en": "Restaurants",
"prepration_time": 15,
"longitude_y": 0.7544,
"longitude_x": -67.2,
"OpeningTime2": "-",
"vendor_tag_name": "Arabic,Breakfast,Burgers,Desserts,Free Delivery,Grills,Lebanese,Salads,Sandwiches,Shawarma",
"status_y": 1,
"status_x": {
 "$binary": "MQ==",
 "$type": "0"
},
"tuesday_to_time2": "23:59:00",
"tuesday_to_time1": "00:30:00",
"customer_id": "TCHWPBT",
"friday_from_time2": "10:00:00",
"friday_from_time1": "00:00:00",
"sunday_to_time2": "23:59:00",
"sunday_to_time1": "00:30:00",
"rank": 11,
"location_number": 0,
```

"tuesday_from_time2": "08:00:00",
"tuesday_from_time1": "00:00:00",
"language": "EN",
"one_click_vendor": true
}

Schema loaded in pgAdmin UI



- **PostgreSQL Schema**

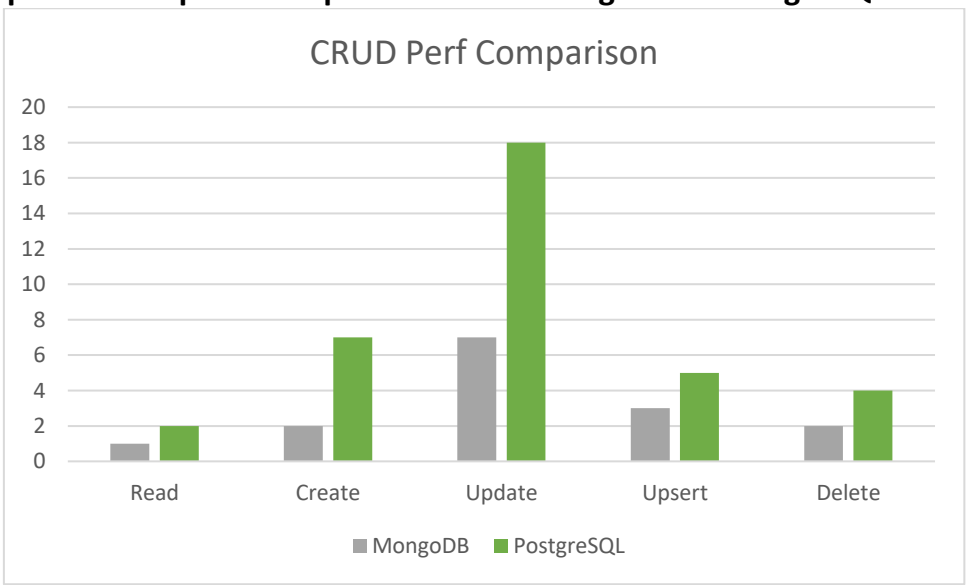
```
CREATE TABLE train_full (
        customer_id VARCHAR,
        gender VARCHAR,
        status_x NUMERIC,
        verified_x NUMERIC,
        created_at_x TIMESTAMP,
        updated_at_x TIMESTAMP,
        location_number NUMERIC,
        location_type VARCHAR,
        latitude_x FLOAT8,
        longitude_x FLOAT8,
        id NUMERIC,
        authentication_id NUMERIC ,
        latitude_y FLOAT8,
        longitude_y FLOAT8,
        vendor_category_en VARCHAR,
        vendor_category_id NUMERIC,
        delivery_charge FLOAT8,
        serving_distance NUMERIC,
        is_open NUMERIC,
        OpeningTime VARCHAR,
        OpeningTime2 VARCHAR,
        prepration_time NUMERIC,
        commission  NUMERIC,
```

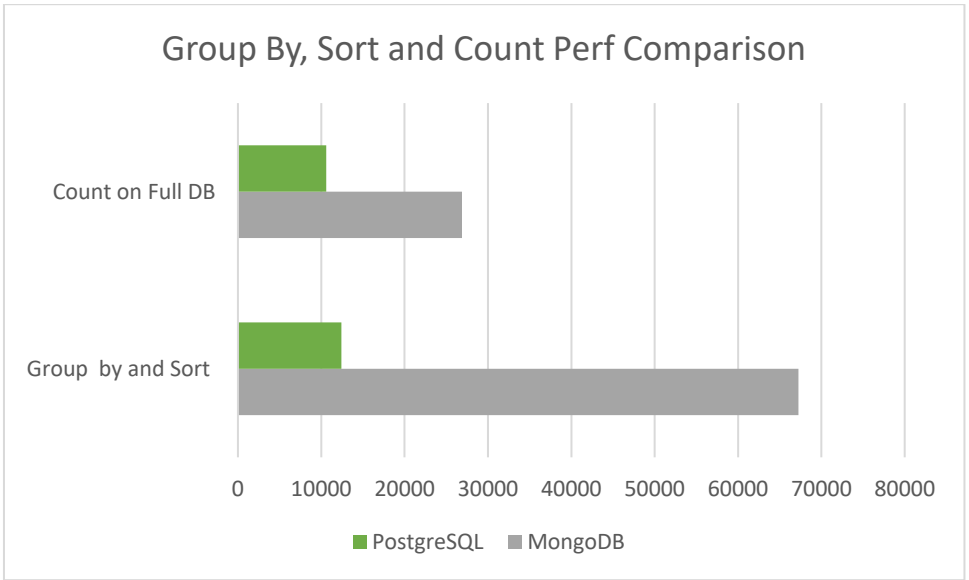
```
is_akeed_delivering VARCHAR,
discount_percentage NUMERIC,
status_y NUMERIC,
verified_y NUMERIC ,
rank NUMERIC,
language VARCHAR,
vendor_rating FLOAT8,
sunday_from_time1 TIME,
sunday_to_time1 TIME,
sunday_from_time2 TIME,
sunday_to_time2 TIME,
monday_from_time1  TIME,
monday_to_time1 TIME ,
monday_from_time2 TIME,
monday_to_time2 TIME,
tuesday_from_time1 TIME,
tuesday_to_time1 TIME,
tuesday_from_time2 TIME,
tuesday_to_time2 TIME,
wednesday_from_time1 TIME,
wednesday_to_time1 TIME,
wednesday_from_time2 TIME,
wednesday_to_time2 TIME,
thursday_from_time1 TIME,
thursday_to_time1 TIME,
thursday_from_time2 TIME,
thursday_to_time2 TIME,
friday_from_time1 TIME,
friday_to_time1 TIME,
friday_from_time2 TIME,
friday_to_time2 TIME,
saturday_from_time1 TIME,
saturday_to_time1 TIME,
saturday_from_time2 TIME,
saturday_to_time2 TIME,
primary_tags VARCHAR,
open_close_flags NUMERIC,
vendor_tag VARCHAR,
vendor_tag_name VARCHAR,
one_click_vendor VARCHAR,
country_id NUMERIC,
city_id NUMERIC,
created_at_y TIMESTAMP,
updated_at_y TIMESTAMP,
device_type NUMERIC ,
display_orders NUMERIC,
location_number_obj NUMERIC,
id_obj NUMERIC,
CID_X_LOC_NUM_X_VENDOR VARCHAR PRIMARY KEY,
target NUMERIC);
```

Tabular Summary and Visualization

DBs	Read (ms)	Create (ms)	Update (ms)	Upsert (ms)	Delete (ms)	Group by and Sort (ms)	Count on Full DB (ms)
MongoDB	1	2	7	3	2	67237	26888
PostgreSQL	2	7	18	5	4	12406	10613

Visual Comparison of operations performed on MongoDB vs PostgreSQL DB





Performance Summary:

- Initial Data load from given source took 85 minutes for mongoDB and it took only 12 minutes for PostgreSQL. This was done using inbuilt native import utility provided by native DBs. So data migration would comparatively take more time for NoSQL DB compared to RDBMS where data model is predefined.

- As we observe from summary table all CRUD operations executed on MongoDB outperform PostgreSQL DB; so MongoDB seems to perform better on CRUD operations. However, being a document based database Groupby, Sort and Counts seem to take more time on MongoDB.

Recommendation:

Hence on basis of CRUD operations, recommendation would be to select MongoDB. Because these operations are performed often in concurrent user environment. Sort and Group are used in less frequent manner and we can fine tune these operations in different ways.

CRUD and other operations on MongoDB

Install dependencies to connect to mongo db (pymongo)

In [1]:
```
pip install pymongo
```

```
Requirement already satisfied: pymongo in c:\users\ram\anaconda3\lib\site-p
ackages (4.0.1)
Note: you may need to restart the kernel to use updated packages.
```

Import required libraries

In [2]:
```
# importing the required libraries
import pymongo
import pprint
import json
import warnings
import time
import pprint
warnings.filterwarnings('ignore')
```

Get connection and db

In [3]:
```
# connect to the mongoclient
client = pymongo.MongoClient('mongodb://localhost:27017')

# get the database
db = client['restaurant']
collection = db['train_full']
```

Execute a find statement based on unique key and capture execution duration

READ or FIND or SELECT

In [7]:
```
## Time taken to find the record
#Start time
t0 = time.time()

#execute query and fetch records
record = collection.find( { "CID X LOC_NUM X VENDOR": "TCHWPBT X 0 X 13" } )

#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time(ms): ' + str(milliseconds))

print('Query Result:')
#print the record
for x in record:
    pprint.pprint(x)
```

```
Execution Time(ms): 1
Query Result:
{'CID X LOC_NUM X VENDOR': 'TCHWPBT X 0 X 13',
 'OpeningTime': '08:30AM-10:30PM',
 'OpeningTime2': '-',
 '_id': ObjectId('61ca2a90b05fde0091bc3ffb'),
 'authentication_id': 118608.0,
 'city_id': 1,
 'commission': 0.0,
 'country_id': 1,
 'created_at_x': '2018-02-07 19:16:23',
 'created_at_y': '2018-05-03 12:32:06',
 'customer_id': 'TCHWPBT',
 'delivery_charge': 0.7,
 'device_type': 3,
 'discount_percentage': 0.0,
 'display_orders': 1,
 'friday_from_time1': '00:00:00',
 'friday_from_time2': '08:00:00',
 'friday_to_time1': '01:30:00',
 'friday_to_time2': '23:59:00',
 'gender': 'Male',
 'id': 13,
 'id_obj': 13,
 'is_akeed_delivering': True,
 'is_open': True,
 'language': 'EN',
 'latitude_x': -96.44,
 'latitude_y': -0.4717,
 'location_number': 0,
 'location_number_obj': 0,
```

```
 'location_type': 'Work',
 'longitude_x': -67.2,
 'longitude_y': 0.7446,
 'monday_from_time1': '00:00:00',
 'monday_from_time2': '08:00:00',
 'monday_to_time1': '01:30:00',
 'monday_to_time2': '23:59:00',
 'one_click_vendor': True,
 'open_close_flags': 1.0,
 'prepration_time': 14,
 'primary_tags': '{"primary_tags":"7"}',
 'rank': 11,
 'saturday_from_time1': '00:00:00',
 'saturday_from_time2': '08:00:00',
 'saturday_to_time1': '01:30:00',
 'saturday_to_time2': '23:59:00',
 'serving_distance': 5.0,
 'status_x': b'1',
 'status_y': 1.0,
 'sunday_from_time1': '00:00:00',
 'sunday_from_time2': '08:00:00',
 'sunday_to_time1': '01:30:00',
 'sunday_to_time2': '23:59:00',
 'target': 0,
 'thursday_from_time1': '00:00:00',
 'thursday_from_time2': '08:00:00',
 'thursday_to_time1': '01:30:00',
 'thursday_to_time2': '19:30:00',
 'tuesday_from_time1': '00:00:00',
 'tuesday_from_time2': '08:00:00',
 'tuesday_to_time1': '01:30:00',
 'tuesday_to_time2': '23:59:00',
 'updated_at_x': '2018-02-07 19:16:23',
 'updated_at_y': '2020-04-05 20:46:03',
 'vendor_category_en': 'Restaurants',
 'vendor_category_id': 2.0,
 'vendor_rating': 4.7,
 'vendor_tag': '4,41,51,34,27,15,24,16,28',
 'vendor_tag_name': 'Breakfast,Cakes,Crepes,Italian,Pasta,Pizzas,Salads,San
dwiches,Soups',
 'verified_x': b'1',
 'verified_y': b'1',
 'wednesday_from_time1': '00:00:00',
 'wednesday_from_time2': '08:00:00',
 'wednesday_to_time1': '01:30:00',
 'wednesday_to_time2': '19:30:00'}
```


Explain information on the execution

In [8]:
```
record = db.train_full.find( { "CID X LOC_NUM X VENDOR": "TCHWPBT X 0 X 13" } )

#print the record
pprint.pprint(record)
```

```
{'command': {'$db': 'restaurant',
             'filter': {'CID X LOC_NUM X VENDOR': 'TCHWPBT X 0 X 13'},
             'find': 'train_full'},
 'executionStats': {'allPlansExecution': [],
                    'executionStages': {'advanced': 1,
                                        'alreadyHasObj': 0,
                                        'docsExamined': 1,
                                        'executionTimeMillisEstimate': 0,
                                        'inputStage': {'advanced': 1,
                                                       'direction': 'forwar
d',
                                                       'dupsDropped': 0,
                                                       'dupsTested': 0,
                                                       'executionTimeMillis
Estimate': 0,
                                                       'indexBounds': {'CID
X LOC_NUM X VENDOR': ['["TCHWPBT '

'X '

'0 '

'X '

'13", '

'"TCHWPBT '

'X '

'0 '

'X '

'13"]']},
                                                       'indexName': 'CID X
'

                                                                    'LOC_NU
M X '
                                                                    'VENDOR

_1',
                                                       'indexVersion': 2,
                                                       'isEOF': 1,
                                                       'isMultiKey': False,
                                                       'isPartial': False,
                                                       'isSparse': False,
                                                       'isUnique': True,
                                                       'keyPattern': {'CID
X LOC_NUM X VENDOR': 1},
                                                       'keysExamined': 1,
                                                       'multiKeyPaths': {'C
```

```
ID X LOC_NUM X VENDOR': []},
                                                        'nReturned': 1,
                                                        'needTime': 0,
                                                        'needYield': 0,
                                                        'restoreState': 0,
                                                        'saveState': 0,
                                                        'seeks': 1,
                                                        'stage': 'IXSCAN',
                                                        'works': 2},
                                               'isEOF': 1,
                                               'nReturned': 1,
                                               'needTime': 0,
                                               'needYield': 0,
                                               'restoreState': 0,
                                               'saveState': 0,
                                               'stage': 'FETCH',
                                               'works': 2},
                         'executionSuccess': True,
                         'executionTimeMillis': 2,
                         'nReturned': 1,
                         'totalDocsExamined': 1,
                         'totalKeysExamined': 1},
 'explainVersion': '1',
 'ok': 1.0,
 'queryPlanner': {'indexFilterSet': False,
                  'maxIndexedAndSolutionsReached': False,
                  'maxIndexedOrSolutionsReached': False,
                  'maxScansToExplodeReached': False,
                  'namespace': 'restaurant.train_full',
                  'parsedQuery': {'CID X LOC_NUM X VENDOR': {'$eq': 'TCHWPB
T X '
                                                                  '0 X 1
3'}},
                  'rejectedPlans': [],
                  'winningPlan': {'inputStage': {'direction': 'forward',
                                                 'indexBounds': {'CID X LOC
_NUM X VENDOR': ['["TCHWPBT '

'X '

'0 '

'X '

'13", '

'"TCHWPBT '

'X '

'0 '

'X '

'13"]']},
                                                 'indexName': 'CID X LOC_NU
M X '
```

```
                                                                'VENDOR_1',
                                                    'indexVersion': 2,
                                                    'isMultiKey': False,
                                                    'isPartial': False,
                                                    'isSparse': False,
                                                    'isUnique': True,
                                                    'keyPattern': {'CID X LOC_
NUM X VENDOR': 1},
                                                    'multiKeyPaths': {'CID X L
OC_NUM X VENDOR': []},
                                                    'stage': 'IXSCAN'},
                                        'stage': 'FETCH'}},
 'serverInfo': {'gitVersion': 'd65fd89df3fc039b5c55933c0f71d647a54510ae',
                'host': 'DESKTOP-F81KN39',
                'port': 27017,
                'version': '5.0.5'},
 'serverParameters': {'internalDocumentSourceGroupMaxMemoryBytes': 10485760
0,
                      'internalDocumentSourceSetWindowFieldsMaxMemoryByte
s': 104857600,
                      'internalLookupStageIntermediateDocumentMaxSizeByte
s': 104857600,
                      'internalQueryFacetBufferSizeBytes': 104857600,
                      'internalQueryFacetMaxOutputDocSizeBytes': 104857600,
                      'internalQueryMaxAddToSetBytes': 104857600,
                      'internalQueryMaxBlockingSortMemoryUsageBytes': 10485
7600,
                      'internalQueryProhibitBlockingMergeOnMongoS': 0}}
```


CREATE or INSERT

In [9]:

```
createRec = {'CID X LOC_NUM X VENDOR': 'Bits_Rec1',
 'OpeningTime': '08:30AM-10:30PM',
 'OpeningTime2': '-',
 #'_id': ObjectId('61ca2a90b05fde0091bc3ffb'),
 'authentication_id': 118608.0,
 'city_id': 1,
 'commission': 0.0,
 'country_id': 1,
 'created_at_x': '2018-02-07 19:16:23',
 'created_at_y': '2018-05-03 12:32:06',
 'customer_id': 'bits_cust',
 'delivery_charge': 0.7,
 'device_type': 3,
 'discount_percentage': 0.0,
 'display_orders': 1,
 'friday_from_time1': '00:00:00',
 'friday_from_time2': '08:00:00',
 'friday_to_time1': '01:30:00',
 'friday_to_time2': '23:59:00',
 'gender': 'Male',
 'id': 13,
 'id_obj': 13,
 'is_akeed_delivering': True,
 'is_open': True,
 'language': 'EN',
 'latitude_x': -96.44,
 'latitude_y': -0.4717,
 'location_number': 0,
 'location_number_obj': 0,
 'location_type': 'Work',
 'longitude_x': -67.2,
 'longitude_y': 0.7446,
 'monday_from_time1': '00:00:00',
 'monday_from_time2': '08:00:00',
 'monday_to_time1': '01:30:00',
 'monday_to_time2': '23:59:00',
 'one_click_vendor': True,
 'open_close_flags': 1.0,
 'prepration_time': 14,
 'primary_tags': '{"primary_tags":"7"}',
 'rank': 11,
 'saturday_from_time1': '00:00:00',
 'saturday_from_time2': '08:00:00',
 'saturday_to_time1': '01:30:00',
 'saturday_to_time2': '23:59:00',
 'serving_distance': 5.0,
 'status_x': b'1',
 'status_y': 1.0,
 'sunday_from_time1': '00:00:00',
 'sunday_from_time2': '08:00:00',
 'sunday_to_time1': '01:30:00',
 'sunday_to_time2': '23:59:00',
 'target': 0,
 'thursday_from_time1': '00:00:00',
 'thursday_from_time2': '08:00:00',
 'thursday_to_time1': '01:30:00',
```

```
 'thursday_to_time2': '19:30:00',
 'tuesday_from_time1': '00:00:00',
 'tuesday_from_time2': '08:00:00',
 'tuesday_to_time1': '01:30:00',
 'tuesday_to_time2': '23:59:00',
 'updated_at_x': '2018-02-07 19:16:23',
 'updated_at_y': '2020-04-05 20:46:03',
 'vendor_category_en': 'Restaurants',
 'vendor_category_id': 2.0,
 'vendor_rating': 4.7,
 'vendor_tag': '4,41,51,34,27,15,24,16,28',
 'vendor_tag_name': 'Breakfast,Cakes,Crepes,Italian,Pasta,Pizzas,Salads,Sandw
 'verified_x': b'1',
 'verified_y': b'1',
 'wednesday_from_time1': '00:00:00',
 'wednesday_from_time2': '08:00:00',
 'wednesday_to_time1': '01:30:00',
 'wednesday_to_time2': '19:30:00'}


## Time taken to find and print the results
#Start time
t0 = time.time()

collection.insert_one(createRec)

#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time (ms): ' + str(milliseconds))
```

```
Execution Time (ms): 25
```

UPDATE only

In [10]:
```
myquery = { 'CID X LOC_NUM X VENDOR': 'Bits_Rec1' }
newvalues = { "$set": { 'customer_id': 'bits_cust11'}}

#Start time
t0 = time.time()

collection.update_one(myquery, newvalues)


#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time(ms) : ' + str(milliseconds))
```

```
Execution Time(ms) : 7
```

UPDATE or INSERT if record is not available (UPSERT)

In [11]:
```
myquery = { 'CID X LOC_NUM X VENDOR': 'Bits_Rec1' }
newvalues = { "$set": { 'customer_id': 'bits_cust11'}}

## Time taken to find and print the results
#Start time
t0 = time.time()

collection.update_one(myquery, newvalues, upsert=True)

#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time(ms) : ' + str(milliseconds))
```

```
Execution Time(ms) : 3
```

DELETE

In [12]:
```
myquery = { 'CID X LOC_NUM X VENDOR': 'Bits_Rec1' }

#Start time
t0 = time.time()

collection.delete_one(myquery)

#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time (ms): ' + str(milliseconds))
```

```
Execution Time (ms): 4
```

GROUP BY and SORT

In [13]:

```
myquery =[{
            "$group":
            {
             "_id": "$vendor_category_en",
             "avgservdist": { "$avg": "$serving_distance" }
            }
           },
                      {
         "$sort": { "avgservdist": 1 }
         }]

#Start time
t0 = time.time()

agg_result= collection.aggregate(
    myquery)

#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time (ms): ' + str(milliseconds))
print('Query Result:')

for i in agg_result:
    print(i)
```

```
Execution Time (ms): 67237
Query Result:
{'_id': 'Restaurants', 'avgservdist': 11.590909090909092}
{'_id': 'Sweets & Bakes', 'avgservdist': 12.916666666666666}
```

COUNT all documents

In [14]:

```
#Start time
t0 = time.time()

doc_count = collection.count_documents({})
#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time (ms): ' + str(milliseconds))

print("Total documents:" + str(doc_count))
```

```
Execution Time (ms): 26888
Total documents:5802400
```

CRUD and other operations on PostgreSQL DB

Install dependencies

In [1]:
```
pip install psycopg2
```

```
Requirement already satisfied: psycopg2 in c:\users\ram\anaconda3\lib\site-
packages (2.9.2)
Note: you may need to restart the kernel to use updated packages.
```

Import libraries ¶

In [2]:
```
import psycopg2
import time
import pprint
```

Make DB Connection and get cursor

In [10]:
```
conn = psycopg2.connect(
    host="localhost",
    database="restaurant",
    user='postgres',
    password='admin',
    port=5432)
cursor = conn.cursor()
```

Execute a SELECT statement based on unique key and capture execution duration

READ or FIND or SELECT

In [6]:

```
## Time taken to find the record
#prepare query
sql_query = "select * from train_full where CID_X_LOC_NUM_X_VENDOR = 'TCHWPBT

#Start time
t0 = time.time()

#execute query and fetch records
cursor.execute(sql_query)
record = cursor.fetchall()

#End Time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

print('Execution Time (ms): ' + str(milliseconds))

print('Query Result:')
#print the record
for x in record:
    pprint.pprint(x)
```

```
Execution Time (ms): 2
Query Result:
('TCHWPBT',
 'Male',
 Decimal('1'),
 Decimal('1'),
 datetime.datetime(2018, 2, 7, 19, 16, 23),
 datetime.datetime(2018, 2, 7, 19, 16, 23),
 Decimal('0'),
 'Work',
 -96.44,
 -67.2,
 Decimal('13'),
 Decimal('118608.0'),
 -0.4717,
 0.7446,
 'Restaurants',
 Decimal('2.0'),
 0.7,
 Decimal('5.0'),
 Decimal('1.0'),
 '08:30AM-10:30PM',
 '-',
 Decimal('14'),
 Decimal('0.0'),
 'Yes',
 Decimal('0.0'),
 Decimal('1.0'),
 Decimal('1'),
```

```
Decimal('11'),
'EN',
4.7,
datetime.time(0, 0),
datetime.time(1, 30),
datetime.time(8, 0),
datetime.time(23, 59),
datetime.time(0, 0),
datetime.time(1, 30),
datetime.time(8, 0),
datetime.time(23, 59),
datetime.time(0, 0),
datetime.time(1, 30),
datetime.time(8, 0),
datetime.time(23, 59),
datetime.time(0, 0),
datetime.time(1, 30),
datetime.time(8, 0),
datetime.time(19, 30),
datetime.time(0, 0),
datetime.time(1, 30),
datetime.time(8, 0),
datetime.time(19, 30),
datetime.time(0, 0),
datetime.time(1, 30),
datetime.time(8, 0),
datetime.time(23, 59),
datetime.time(0, 0),
datetime.time(1, 30),
datetime.time(8, 0),
datetime.time(23, 59),
'{primary_tags:7}',
Decimal('1.0'),
'4,41,51,34,27,15,24,16,28',
'Breakfast,Cakes,Crepes,Italian,Pasta,Pizzas,Salads,Sandwiches,Soups',
'Y',
Decimal('1.0'),
Decimal('1.0'),
datetime.datetime(2018, 5, 3, 12, 32, 6),
datetime.datetime(2020, 4, 5, 20, 46, 3),
Decimal('3'),
Decimal('1'),
Decimal('0'),
Decimal('13'),
'TCHWPBT X 0 X 13',
Decimal('0'))
```

Explain information on the execution

In [7]:
```
#prepare query
sql_queryexplain = "explain analyze select * from train_full where CID_X_LOC_

#execute query and fetch records
cursor.execute(sql_queryexplain)
record = cursor.fetchall()

#print the record
pprint.pprint(record)
```

```
[('Index Scan using train_full_pkey on train_full  (cost=0.56..8.57 rows=1
'
  'width=554) (actual time=0.069..0.070 rows=1 loops=1)',),
 ("  Index Cond: ((cid_x_loc_num_x_vendor)::text = 'TCHWPBT X 0 X 13'::tex
t)",),
 ('Planning Time: 0.264 ms',),
 ('Execution Time: 0.145 ms',)]
```

CREATE or INSERT

In [11]:
```
query = """INSERT INTO train_full(customer_id, cid_x_loc_num_x_vendor) VALUES
data = ('Bits_cust','bits_rec2')


#Start time
t0 = time.time()

cursor.execute(query, data)
conn.commit()

#End Time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

print('Execution Time (ms): ' + str(milliseconds))
```

```
Execution Time (ms): 7
```

UPDATE only

In [12]:
```
#Start time
t0 = time.time()

cursor.execute("Update train_full set customer_id = ('Bits_custupd') where ci
conn.commit()
#End Time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

print('Execution Time (ms): ' + str(milliseconds))
```

```
Execution Time (ms): 18
```

UPDATE or INSERT if record is not available (UPSERT)

In [15]:
```
insert_sql = '''INSERT INTO train_full (customer_id, cid_x_loc_num_x_vendor)
data = ('Bits_cust','bits_rec1')

#Start time
t0 = time.time()

cursor.execute(insert_sql, data)
conn.commit()

#End Time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

print('Execution Time (ms): ' + str(milliseconds))
```

```
Execution Time (ms): 2
```

DELETE

In [16]:
```
#Start time
t0 = time.time()

cursor.execute("DELETE FROM train_full where cid_x_loc_num_x_vendor = 'bits_r
conn.commit()
#End Time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

print('Execution Time (ms): ' + str(milliseconds))
```

```
Execution Time (ms): 4
```

GROUP BY and SORT

In [17]:
```
query = """SELECT vendor_category_en, AVG (serving_distance) AS serving_dista

#Start time
t0 = time.time()

cursor.execute(query)
record = cursor.fetchall()
#End Time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

print('Execution Time (ms): ' + str(milliseconds))

print('Query Result:')
#print the record
for x in record:
    pprint.pprint(x)
```

```
Execution Time (ms): 12406
Query Result:
('Restaurants', Decimal('11.5909090909090909'))
('Sweets & Bakes', Decimal('12.9166666666666667'))
(None, None)
```

Count all records

In [18]:

```
#Start time
t0 = time.time()

cursor.execute("select count(*) from train_full")
#End time
t1 = time.time()

#Elapsed time
total_n = t1-t0

#convert to milisecs
milliseconds = int(round(total_n * 1000))

#print execution time
print('Execution Time (ms): ' + str(milliseconds))

result = cursor.fetchone()
for r1 in result:
  print("Total Records: " + str(r1))
```

```
Execution Time (ms): 10613
Total Records: 5802401
```