1 MongoDB

Q1

a)

```
> db.restaurants.find({
    borough: {
      $in: ["Bronx", "Brooklyn"]
    }
  },{_id:0, name:1}).limit(5)
< {
    name: 'Morris Park Bake Shop'
  }
  {
    name: "Wendy'S"
  }
  {
    name: 'Riviera Caterer'
  }
  {
    name: "Wilken'S Fine Food"
  }
  {
    name: 'Regina Caterers'
  }
```

c)

```
> db.restaurants.find({
    borough: {
      $in: ["Bronx", "Brooklyn"]
    }
  }).count()
< 993
```

```
> db.restaurants.find({name:/^Mad/},{_id:1, name:1, borough:1, cuisine:1}).limit(3)
< {
    _id: ObjectId("65569eab735bbc4d443faec6"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Madison Square'
  }
  {
    _id: ObjectId("65569eab735bbc4d443faf94"),
    borough: 'Manhattan',
    cuisine: 'Indian',
    name: 'Madras Mahal'
  }
  {
    _id: ObjectId("65569eac735bbc4d443fb242"),
    borough: 'Manhattan',
    cuisine: 'American ',
    name: 'Madame X'
  }
> db.restaurants.find({name:/^Mad/},{_id:1, name:1, borough:1, cuisine:1}).count()
< 8
```

b)

```
> db.restaurants.find({grades:{$elemMatch:{score:{$gte:80,$lte:90}}}},{_id:0,name:1})
< {
    name: 'B.B. Kings'
  }
  {
    name: 'West 79Th Street Boat Basin Cafe'
  }
```

c)

```
> db.restaurants.find({grades:{$elemMatch:{score:{$gte:80,$lte:90}}}}).count()
< 2
```

```
> db.restaurants.find({grades:{$elemMatch:{grade:'C',date:{
          $gte: ISODate('2014-01-01'),
          $lt: ISODate('2015-01-01')
        }}}},{_id:1,name:1}).limit(3)
< {
    _id: ObjectId("65569eab735bbc4d443fa9c0"),
    name: 'B & M Hot Bagel & Grocery'
  }
  {
    _id: ObjectId("65569eab735bbc4d443fa9c1"),
    name: 'Texas Rotisserie'
  }
  {
    _id: ObjectId("65569eab735bbc4d443fa9ce"),
    name: 'Nyac Main Dining Room'
  }
> db.restaurants.find({grades:{$elemMatch:{grade:'C',date:{
          $gte: ISODate('2014-01-01'),
          $lt: ISODate('2015-01-01')
        }}}},{_id:1,name:1}).count()
< 94
```

d)

e)

```
> db.restaurants.aggregate([
    {
      $group: {
        _id: "$cuisine",
        count: { $sum: 1 }
      }
    },
    {$sort:{count:-1}},
    {$limit:1}
  ])
< {
    _id: 'American ',
    count: 1255
  }
```

f) Getting 8 rows

```
> db.restaurants.aggregate([
    {
  $match: {cuisine:{$ne:'American'}}
  },
    {
      $unwind: "$grades"
    },
    {
      $group: {
        _id: "$restaurant_id",
        average_score: { $avg: "$grades.score" }
      }
    },
    {
  $match:{average_score:{$gt:30}}
  }


  ])
< {
    _id: '40393488',
    average_score: 38.6
  }
  {
    _id: '40756344',
    average_score: 36
  }
```

```
>_MONGOSH
    average_score: 36
  }
  {
    _id: '40387237',
    average_score: 32.6
  }
  {
    _id: '40372466',
    average_score: 33.666666666666664
  }
  {
    _id: '40366157',
    average_score: 32.142857142857146
  }
  {
    _id: '40825993',
    average_score: 30.8
  }
  {
    _id: '40624470',
    average_score: 30.6
  }
  {
    _id: '40374268',
    average_score: 30.8
  }
```

```
> db.restaurants.aggregate([
  {$project: {grades2014Onwards:{$filter:{
 input: '$grades',
 as: 'grade',
 cond: {$gte:["$$grade.date",ISODate('2014-01-01')]}
 }},name:1,_id:0}}
 ])
< {
  name: 'Morris Park Bake Shop',
  grades2014Onwards: [
    {
      date: 2014-03-03T00:00:00.000Z,
      grade: 'A',
      score: 2
    }
  ]
}
{
  name: "Wendy'S",
  grades2014Onwards: [
    {
      date: 2014-12-30T00:00:00.000Z,
      grade: 'A',
      score: 8
    },
```

g)

```
{
  name: "Wendy'S",
  grades2014Onwards: [
    {
      date: 2014-12-30T00:00:00.000Z,
      grade: 'A',
      score: 8
    },
    {
      date: 2014-07-01T00:00:00.000Z,
      grade: 'B',
      score: 23
    }
  ]
}
{
  name: 'Dj Reynolds Pub And Restaurant',
  grades2014Onwards: [
    {
      date: 2014-09-06T00:00:00.000Z,
      grade: 'A',
      score: 2
    }
  ]
}
```

```
          as: 'grade',
          cond: { $gte: ["$$grade.date", ISODate('2014-01-01')] }
        }
      },
      name: 1,
      _id: 0
    }
  },
  {
    $project: {
      countGrades2014Onwards: { $size: "$grades2014Onwards" },
      name: 1
    }
  },
  {
    $group: {
      _id: null,
      totalCount: { $sum: "$countGrades2014Onwards" }
    }
  }
])
< {
  _id: null,
  totalCount: 5470
}
```

```
> db.restaurants.aggregate([
    {
      $unwind: "$grades"
    },
    {
      $group: {
        _id: null,
        averageScore: { $avg: "$grades.score" }
      }
    }
  ])
< {
    _id: null,
    averageScore: 11.427736743468195
  }
```

h)

Q2

```
> db.sales.insertMany([
  { "_id": 1, "city": "Berkeley", "state": "CA", "qty": 648 },
  { "_id": 2, "city": "Bend", "state": "OR", "qty": 491 },
  { "_id": 3, "city": "Kensington", "state": "CA", "qty": 233 },
  { "_id": 4, "city": "Eugene", "state": "OR", "qty": 842 },
  { "_id": 5, "city": "Reno", "state": "NV", "qty": 655 },
  { "_id": 6, "city": "Portland", "state": "OR", "qty": 408 },
  { "_id": 7, "city": "Sacramento", "state": "CA", "qty": 574 }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': 1,
      '1': 2,
      '2': 3,
      '3': 4,
      '4': 5,
      '5': 6,
      '6': 7
    }
  }
```

a)

```
> db.sales.aggregate([
    {$group:{_id:{state:"$state"},total_qty:{$sum:"$qty"}}},
    {$sort:{total_qty:-1}}
  ])
< {
    _id: {
      state: 'OR'
    },
    total_qty: 1741
  }
  {
    _id: {
      state: 'CA'
    },
    total_qty: 1455
  }
  {
    _id: {
      state: 'NV'
    },
    total_qty: 655
  }
```

b) Update operations

i)

```
> db.sales.updateOne({city:'Berkeley'},{$set:{qty:750}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
> db.sales.updateMany({state:'OR'},{$inc:{qty:50}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 3,
    modifiedCount: 3,
    upsertedCount: 0
  }
```

ii)

```
> db.sales.updateOne({_id:5},{$set:{salespeople:['David','Martha']}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

iii)

```
> db.sales.updateOne({_id:5},{$push:{salespeople:'James'}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

iv)

```
{
  _id: 5,
  city: 'Reno',
  state: 'NV',
  qty: 655,
  salespeople: [
    'David',
    'Martha',
    'James'
  ]
}
```

v)

```
> db.sales.updateOne({_id:5,salespeople:'Martha'},{$set:{'salespeople.$':'Lisa'}})
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
```

```
{
  _id: 5,
  city: 'Reno',
  state: 'NV',
  qty: 655,
  salespeople: [
    'David',
    'Lisa',
    'James'
  ]
}
```

vi)

```
> db.sales.deleteMany({state:'CA'})
< {
    acknowledged: true,
    deletedCount: 3
  }
```

```
> db.sales.find({})
< {
    _id: 2,
    city: 'Bend',
    state: 'OR',
    qty: 541
  }
  {
    _id: 4,
    city: 'Eugene',
    state: 'OR',
    qty: 892
  }
  {
    _id: 5,
    city: 'Reno',
    state: 'NV',
    qty: 655,
    salespeople: [
      'David',
      'Lisa',
      'James'
    ]
  }
```

c)

```
  }
  {
    _id: 4,
    city: 'Eugene',
    state: 'OR',
    qty: 892
  }
  {
    _id: 5,
    city: 'Reno',
    state: 'NV',
    qty: 655,
    salespeople: [
      'David',
      'Lisa',
      'James'
    ]
  }
  {
    _id: 6,
    city: 'Portland',
    state: 'OR',
    qty: 458
  }
```

Q3

```
In [1]: from pymongo import MongoClient

In [2]: client = MongoClient("localhost", 27017)
        db = client.booksdb

In [8]: category = input('Enter category')

        pipeline = [
            { "$unwind": '$categories'},
            { "$match": {'categories':category}},
            {"$project":{'title':{"$ifNull":["$title","NA"]},
                         'isbn':{"$ifNull":["$isbn","NA"]},
                         '_id':0}}
        ]

        cursor = db.books.aggregate(pipeline)

        for book in cursor:
            print(f"ISBN: {book['isbn']}, title: {book['title']}")

        Enter categoryComputer Graphics
        ISBN: 1884777902, title: 3D User Interfaces with Java 3D
        ISBN: 133034054, title: Graphics File Formats
        ISBN: 1933988398, title: Gnuplot in Action
        ISBN: 1884777473, title: The Awesome Power of Direct3D/DirectX
        ISBN: 138412146, title: Power-3D
        ISBN: 1930110022, title: Graphics Programming with Perl
```

2 Map Reduce

Q1

Thought process: The map function extracts relevant attributes from each tuple. If that tuple doesn't have that attribute then it should return NIL. It emits the ID of that tuple along with the relevant attribute. The reduce function returns the aggregated attributes with their ID.

```
// Assuming that Tuple is a dictionary or JSON type data type whose attributes can be accessed
// in a similar way. Also assuming that ID attribute of a tuple can be accessed as t[ID]

map (Tuple t, List S):
        for each attribute a in S:
                if t[a] is not NIL:
                        emit(t[ID],t[a])
                else
                        emit(t[ID],NIL)

reduce (Attribute ID, List attributes):
        emit(ID,attributes)
```

Q2

Thought process: The map function maps each tuple to the relation it belongs to. The reduce function returns those tuples that belong to both relations i.e. length of relations list must be 2. The reduce function kind of works like an identity function in this case i.e. it simply passes each key-value pair to the output

```
map (Tuple t, Relation X):
        emit(t,X)

reduce (Tuple t, List relations):
        if (length(relations) == 2):
                emit(t,relations)
```

Q3

Thought process: To group tuples, attribute A is used as key and attribute B is used as values. To avoid errors with the aggregation function, if any tuple doesn't have a B attribute, it is not emitted. In the reduce function the aggregation operation is applied on the B-values and are emitted along with the A attribute.

```
map (Tuple t):
        if t[B] is not NIL:
                emit(t[A],t[B])

reduce (Attribute A, List b_values):
        emit(A, θ(b_values))
```

In [1]:
```python
from pymongo import MongoClient
```

In [2]:
```python
client = MongoClient("localhost", 27017)
db = client.booksdb
```

In [3]:
```python
category = input('Enter category')

pipeline = [
    { "$unwind": '$categories'},
    { "$match": {'categories':category}},
    {"$project":{'title':{"$ifNull":["$title","NA"]},
                 'isbn':{"$ifNull":["$isbn","NA"]},
                 '_id':0}}
]

cursor = db.books.aggregate(pipeline)

for book in cursor:
    print(f"ISBN: {book['isbn']}, title: {book['title']}")
```

```
Enter categoryComputer Graphics
ISBN: 1884777902, title: 3D User Interfaces with Java 3D
ISBN: 133034054, title: Graphics File Formats
ISBN: 1933988398, title: Gnuplot in Action
ISBN: 1884777473, title: The Awesome Power of Direct3D/DirectX
ISBN: 138412146, title: Power-3D
ISBN: 1930110022, title: Graphics Programming with Perl
```

In [ ]:

In [ ]:

In [1]:
```
!pip install pyspark
```
```
Collecting pyspark
  Downloading pyspark-3.5.0.tar.gz (316.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 316.9/316.9 MB 3.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.0-py2.py3-none-any.whl size=317425344 sha256=f04e5ec94893948e582d8da6d21dcc11c30f048c02e29f281af4e0ea654452e3
  Stored in directory: /root/.cache/pip/wheels/41/4e/10/c2cf2467f71c678cfc8a6b9ac9241e5e44a01940da8fbb17fc
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.0
```

In [2]:
```
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
```
```
The following additional packages will be installed:
  libxtst6 openjdk-8-jre-headless
Suggested packages:
  openjdk-8-demo openjdk-8-source libnss-mdns fonts-dejavu-extra fonts-nanum fonts-ipafont-gothic
  fonts-ipafont-mincho fonts-wqy-microhei fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  libxtst6 openjdk-8-jdk-headless openjdk-8-jre-headless
0 upgraded, 3 newly installed, 0 to remove and 9 not upgraded.
Need to get 39.7 MB of archives.
After this operation, 144 MB of additional disk space will be used.
Selecting previously unselected package libxtst6:amd64.
(Reading database ... 120880 files and directories currently installed.)
Preparing to unpack .../libxtst6_2%3a1.2.3-1build4_amd64.deb ...
Unpacking libxtst6:amd64 (2:1.2.3-1build4) ...
Selecting previously unselected package openjdk-8-jre-headless:amd64.
Preparing to unpack .../openjdk-8-jre-headless_8u382-ga-1~22.04.1_amd64.deb ...
Unpacking openjdk-8-jre-headless:amd64 (8u382-ga-1~22.04.1) ...
Selecting previously unselected package openjdk-8-jdk-headless:amd64.
Preparing to unpack .../openjdk-8-jdk-headless_8u382-ga-1~22.04.1_amd64.deb ...
Unpacking openjdk-8-jdk-headless:amd64 (8u382-ga-1~22.04.1) ...
Setting up libxtst6:amd64 (2:1.2.3-1build4) ...
Setting up openjdk-8-jre-headless:amd64 (8u382-ga-1~22.04.1) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/orbd to provide /usr/bin/orbd (orbd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/servertool to provide /usr/bin/servertool (servertool) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/tnameserv to provide /usr/bin/tnameserv (tnameserv) in auto mode
Setting up openjdk-8-jdk-headless:amd64 (8u382-ga-1~22.04.1) ...
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/clhsdb to provide /usr/bin/clhsdb (clhsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/extcheck to provide /usr/bin/extcheck (extcheck) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/hsdb to provide /usr/bin/hsdb (hsdb) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/idlj to provide /usr/bin/idlj (idlj) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/javah to provide /usr/bin/javah (javah) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jhat to provide /usr/bin/jhat (jhat) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/jsadebugd to provide /usr/bin/jsadebugd (jsadebugd) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/native2ascii to provide /usr/bin/native2ascii (native2ascii) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/schemagen to provide /usr/bin/schemagen (schemagen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsgen to provide /usr/bin/wsgen (wsgen) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/wsimport to provide /usr/bin/wsimport (wsimport) in auto mode
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/bin/xjc to provide /usr/bin/xjc (xjc) in auto mode
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link
```

In [3]:
```
import os
```

In [4]:
```
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
```

In [5]:
```
from pyspark import SparkContext
```

In [6]:
```
sc = SparkContext('local','My App')
sc
```

Out[6]:

SparkContext

Spark UI

Version	v3.5.0
Master	local
AppName	My App

In [7]:
```
from google.colab import drive
drive.mount('/content/drive')
```
```
Mounted at /content/drive
```

In [8]:
```
import time
```

In [9]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("JSON RDD Operations").getOrCreate()
```

In [10]:
```
path = "/content/drive/MyDrive/IDMP Assignment 4 files/Bombing_Operations.json"
```

In [11]:
```
# Q1 (Restarted run time before executing each operation)
```

In [12]:
```
# With RDD operations
```

In [13]:
```
json_rdd = spark.read.json(path).rdd
```

In [14]:
```
start_time_rdd = time.time()
grouped_rdd = json_rdd.map(lambda x: (x['ContryFlyingMission'], 1)).reduceByKey(lambda a, b: a + b)

result = grouped_rdd.collect()
for item in result:
    print(f"Country: {item[0]}, Count: {item[1]}")

end_time_rdd = time.time()
execution_time_rdd = end_time_rdd - start_time_rdd
print(f"RDD Operation Execution Time: {execution_time_rdd} seconds")
```
```
Country: UNITED STATES OF AMERICA, Count: 3708997
Country: AUSTRALIA, Count: 12519
Country: LAOS, Count: 32777
Country: KOREA (SOUTH), Count: 24469
Country: VIETNAM (SOUTH), Count: 622013
RDD Operation Execution Time: 96.90672063827515 seconds
```

In [11]:
```
# With the DataFrame API
```

In [12]:
```
json_df = spark.read.json(path)
```

In [13]:
```
json_df.show()
```
```
+--------+--------------------+-----------+-------------------+-----------+---------------+-------------+------------+--------------------+-------------------+
|AirCraft| ContryFlyingMission|MissionDate|OperationSupported|PeriodOfDay|TakeoffLocation|TargetCountry|TimeOnTarget|          WeaponType|WeaponsLoadedWeight|
+--------+--------------------+-----------+-------------------+-----------+---------------+-------------+------------+--------------------+-------------------+
|   EC-47|UNITED STATES OF ...| 1971-06-05|              NULL|          D|   TAN SON NHUT|     CAMBODIA|      1005.0|                NULL|                  0|
|   EC-47|UNITED STATES OF ...| 1972-12-26|              NULL|          D|  NAKHON PHANOM|SOUTH VIETNAM|       530.0|                NULL|                  0|
|    RF-4|UNITED STATES OF ...| 1973-07-28|              NULL|          D|       UDORN AB|         LAOS|       730.0|                NULL|                  0|
|     A-1|UNITED STATES OF ...| 1970-02-02|              NULL|          N|  NAKHON PHANOM|         LAOS|      1415.0|BLU27 FIRE BOMB (...|              17400|
|    A-37|     VIETNAM (SOUTH)| 1970-10-08|              NULL|          D|         DANANG|SOUTH VIETNAM|      1240.0|                NULL|                  0|
|     F-4|UNITED STATES OF ...| 1970-11-25|              NULL|          D|        UBON AB|         LAOS|       650.0|MK 82 GP BOMB (50...|              31860|
|     A-4|UNITED STATES OF ...| 1972-03-08|              NULL|          D|    TONKIN GULF|         LAOS|      1005.0|                NULL|                  0|
|     F-4|UNITED STATES OF ...| 1971-12-27|              NULL|       NULL|       UDORN AB|         LAOS|         0.0|                NULL|                  0|
|     A-7|UNITED STATES OF ...| 1972-05-24|              NULL|       NULL|    TONKIN GULF|NORTH VIETNAM|         0.0|                NULL|                  0|
|   EC-47|UNITED STATES OF ...| 1972-09-12|              NULL|          D|   TAN SON NHUT|SOUTH VIETNAM|       710.0|                NULL|                  0|
|   CH-53|UNITED STATES OF ...| 1974-06-13|              NULL|          N|  NAKHON PHANOM|     THAILAND|      1800.0|                NULL|                  0|
|   CH-53|UNITED STATES OF ...| 1974-12-19|              NULL|          D|  NAKHON PHANOM|     THAILAND|       800.0|                NULL|                  0|
|     O-1|     VIETNAM (SOUTH)| 1973-10-24|              NULL|          D|      NHA TRANG|SOUTH VIETNAM|       800.0|                NULL|                  0|
|    UH-1|     VIETNAM (SOUTH)| 1974-03-19|              NULL|          D|        PHU CAT|SOUTH VIETNAM|       800.0|        CARGO (TONS)|                  0|
|     C-7|UNITED STATES OF ...| 1970-05-08|              NULL|          D|   TAN SON NHUT|SOUTH VIETNAM|       800.0|                NULL|                  0|
|     A-6|UNITED STATES OF ...| 1971-05-12|              NULL|          N|    TONKIN GULF|         LAOS|      1304.0|CBU24 AN PR/MT (B...|              33200|
|   EB-66|UNITED STATES OF ...| 1971-12-03|              NULL|          N|          KORAT|         LAOS|      1445.0|                NULL|                  0|
|    T-28|                LAOS| 1971-12-19|              NULL|          D|     SAVANAKHET|         LAOS|       230.0|                NULL|                  0|
|     A-6|UNITED STATES OF ...| 1972-08-18|              NULL|       NULL|    TONKIN GULF|NORTH VIETNAM|         0.0|                NULL|                  0|
|     A-7|UNITED STATES OF ...| 1972-10-15|              NULL|          D|    TONKIN GULF|NORTH VIETNAM|       110.0|                NULL|                  0|
+--------+--------------------+-----------+-------------------+-----------+---------------+-------------+------------+--------------------+-------------------+
only showing top 20 rows
```

In [14]:
```
start_time_df = time.time()

json_df.groupBy(json_df['ContryFlyingMission']).count().show()

end_time_df = time.time()
execution_time_df = end_time_df - start_time_df
print(f"DataFrame API Execution Time: {execution_time_df} seconds")
```
```
+--------------------+-------+
| ContryFlyingMission|  count|
+--------------------+-------+
|     VIETNAM (SOUTH)| 622013|
|       KOREA (SOUTH)|  24469|
|UNITED STATES OF ...|3708997|
|           AUSTRALIA|  12519|
|                LAOS|  32777|
+--------------------+-------+

DataFrame API Execution Time: 22.407304048538208 seconds
```

In [12]:
```
# using Spark SQL
```

In [13]:
```
json_df = spark.read.json(path)
json_df.createOrReplaceTempView('operations')
```

In [14]:
```
query = "select ContryFlyingMission, COUNT(*) from operations group by ContryFlyingMission"
```

In [15]:
```
start_time_sql = time.time()

sql_df = spark.sql(query)
sql_df.show()

end_time_sql = time.time()
execution_time_sql = end_time_sql - start_time_sql
print(f"Spark SQL Execution Time: {execution_time_sql} seconds")
```
```
+--------------------+--------+
| ContryFlyingMission|count(1)|
+--------------------+--------+
|     VIETNAM (SOUTH)|  622013|
|       KOREA (SOUTH)|   24469|
|UNITED STATES OF ...| 3708997|
|           AUSTRALIA|   12519|
|                LAOS|   32777|
+--------------------+--------+

Spark SQL Execution Time: 20.657620668411255 seconds
```

In [17]:
```
# Hence, Spark SQL method was the most efficient as it took least amount of time to execute (20.65s) compared to
# DataFrame API (22.40s) and RDD operations (96.90s)
```

In [18]:
```
# Q2 Plot a bar chart with the number of missions by country.
```

In [26]:
```
from pyspark.sql.functions import col
from pyspark.sql.functions import count
```

In [33]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [34]:
```
%matplotlib inline
```

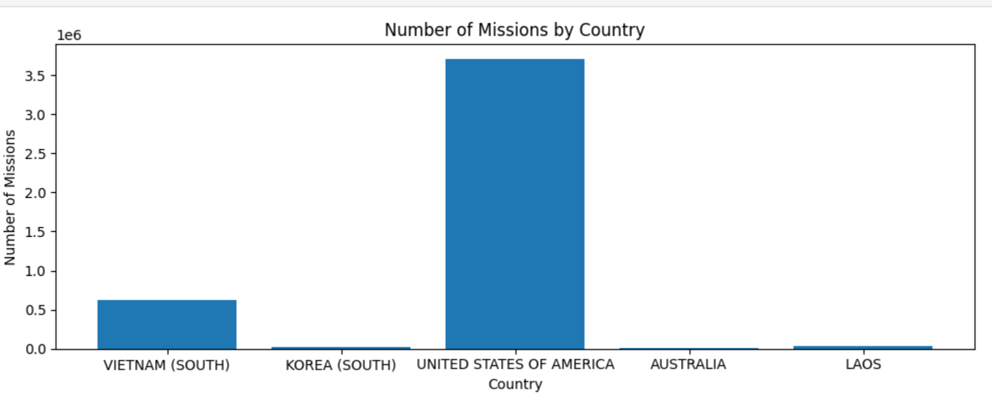
In [22]:
```
# Read JSON File into DataFrame
json_df = spark.read.json(path)
```

In [30]:
```
missions_by_country = json_df.groupBy(json_df['ContryFlyingMission']).count()
```

In [31]:
```
missions_by_country.show()
```
```
+--------------------+-------+
| ContryFlyingMission|  count|
+--------------------+-------+
|     VIETNAM (SOUTH)| 622013|
|       KOREA (SOUTH)|  24469|
|UNITED STATES OF ...|3708997|
|           AUSTRALIA|  12519|
|                LAOS|  32777|
+--------------------+-------+
```

In [37]:
```
# Convert Spark DataFrame to Pandas DataFrame
missions_pd = missions_by_country.toPandas()
```

In [45]:
```
# Plotting
plt.figure(figsize=(10, 4))
plt.bar(missions_pd['ContryFlyingMission'], missions_pd['count'])
plt.xlabel('Country')
plt.ylabel('Number of Missions')
plt.title('Number of Missions by Country')
plt.tight_layout()
plt.show()
```



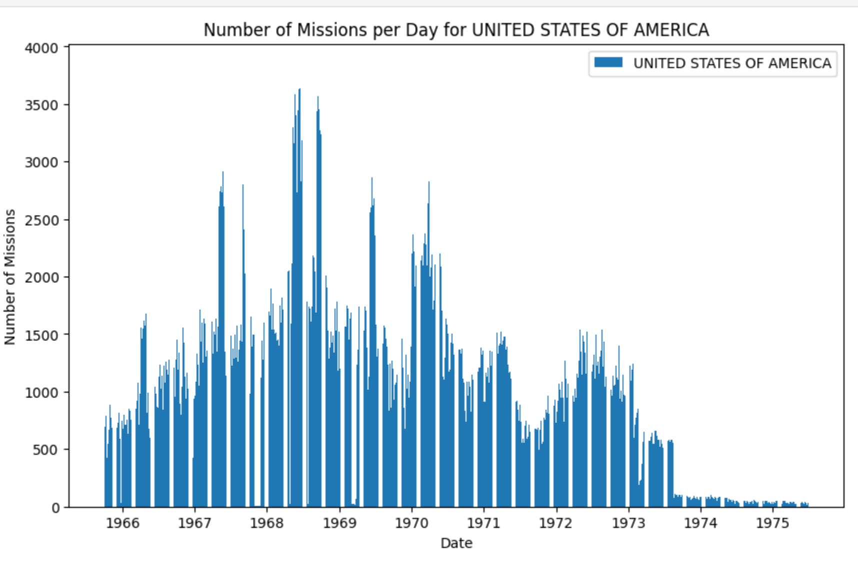
In [46]:
```
# Q3 Plot the number of missions per day for each of the countries involved.
```

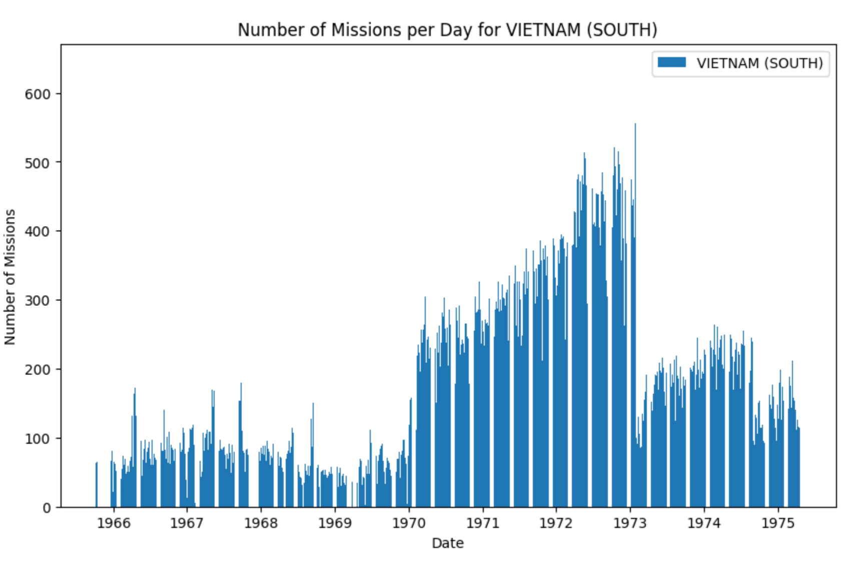
In [53]:
```
missions_per_day = json_df.groupBy(json_df['ContryFlyingMission'],json_df['MissionDate']).count()
```

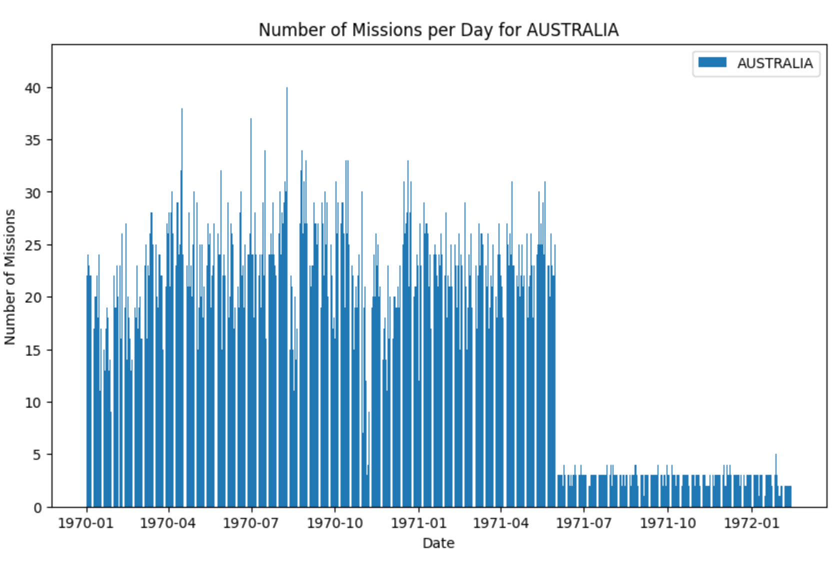
In [54]:
```
missions_per_day.show()
```
```
+--------------------+-----------+-----+
| ContryFlyingMission|MissionDate|count|
+--------------------+-----------+-----+
|UNITED STATES OF ...| 1970-04-08| 1983|
|UNITED STATES OF ...| 1973-02-01|  661|
|     VIETNAM (SOUTH)| 1971-09-10|  371|
|UNITED STATES OF ...| 1973-08-01|  559|
|UNITED STATES OF ...| 1970-12-09| 1259|
|UNITED STATES OF ...| 1970-11-11| 1072|
|           AUSTRALIA| 1970-07-23|   26|
|UNITED STATES OF ...| 1971-03-18| 1342|
|UNITED STATES OF ...| 1974-05-20|   63|
|     VIETNAM (SOUTH)| 1973-05-02|  189|
|     VIETNAM (SOUTH)| 1972-10-30|  486|
|       KOREA (SOUTH)| 1973-11-22|   26|
|UNITED STATES OF ...| 1970-01-04| 1730|
|     VIETNAM (SOUTH)| 1971-09-24|  345|
|     VIETNAM (SOUTH)| 1972-03-07|  408|
|     VIETNAM (SOUTH)| 1970-05-05|  216|
|     VIETNAM (SOUTH)| 1972-07-15|  429|
|                LAOS| 1971-01-09|   23|
|           AUSTRALIA| 1971-03-17|   21|
|           AUSTRALIA| 1970-04-21|   24|
+--------------------+-----------+-----+
only showing top 20 rows
```

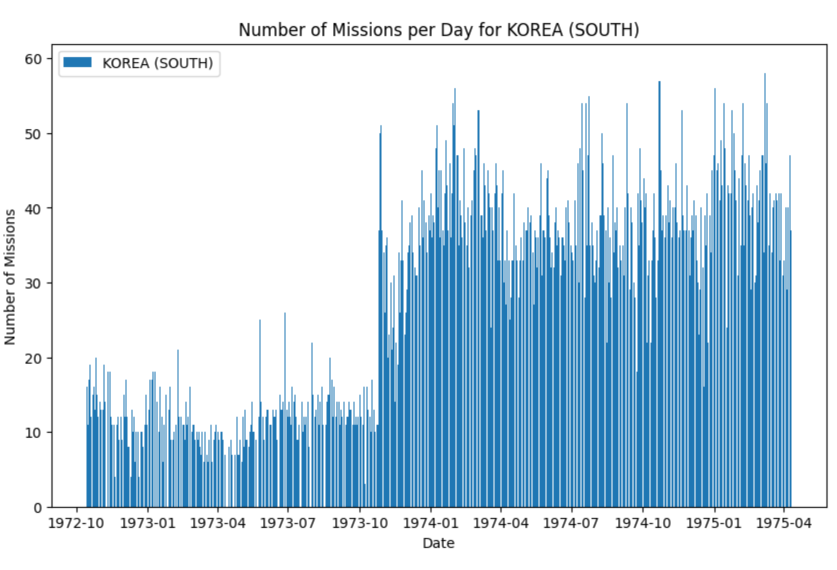
In [55]:
```
missions_pd = missions_per_day.toPandas()
```

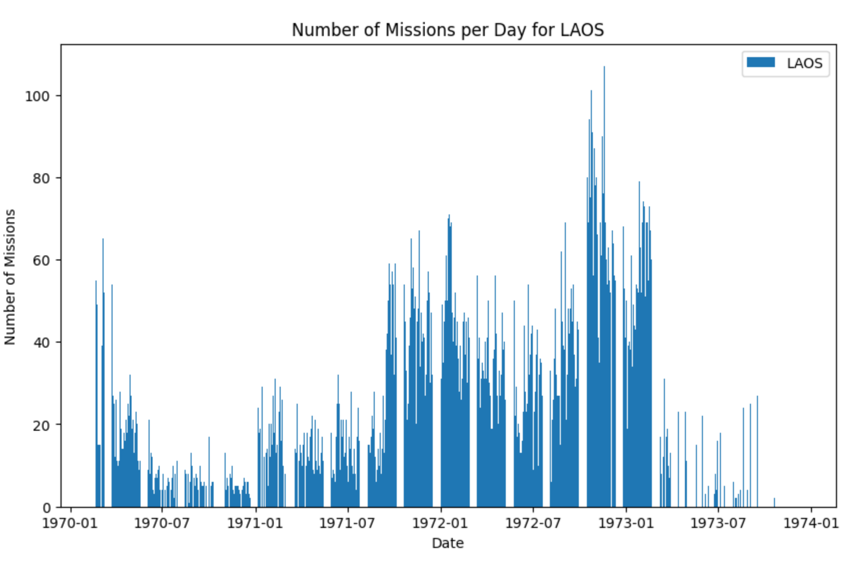
In [58]:
```
countries = missions_pd['ContryFlyingMission'].unique()
for country in countries:
    country_data = missions_pd[missions_pd['ContryFlyingMission'] == country]
    plt.figure(figsize=(10, 6))
    plt.bar(country_data['MissionDate'], country_data['count'], label=country)
    plt.xlabel('Date')
    plt.ylabel('Number of Missions')
    plt.title(f'Number of Missions per Day for {country}')
    plt.legend()
    plt.show()
```











In [60]:
```
# Q4 How many takeoffs were launched to attack North Vietnam on 29 June 1966 from each location?
```

In [61]:
```
json_df = spark.read.json(path)
json_df.createOrReplaceTempView('operations')
```

In [64]:
```
query = """select TakeoffLocation, count(*) from operations
          where MissionDate = '1966-06-29' and
          TargetCountry = 'NORTH VIETNAM'
          group by TakeoffLocation
        """
```

In [65]:
```
sql_df = spark.sql(query)
sql_df.show()
```
```
+----------------+--------+
| TakeoffLocation|count(1)|
+----------------+--------+
|    TAN SON NHUT|      26|
|          DANANG|      35|
|        UDORN AB|      44|
|HANCOCK (CVA-19)|      10|
|   CONSTELLATION|      87|
|          TAKHLI|      56|
|          RANGER|      35|
|           KORAT|      55|
|         UBON AB|      44|
|         CUBI PT|       1|
|    CAM RANH BAY|       2|
+----------------+--------+
```

In [66]:
```
# Q5 Which campaigns saw the heaviest bombings? (the names of the campaigns are stored in OperationSupported).
```

In [84]:
```
query = """select OperationSupported, count(*) as freq from operations
          WHERE OperationSupported IS NOT NULL
          group by OperationSupported
          order by freq DESC
          LIMIT 1
        """
```

In [85]:
```
sql_df = spark.sql(query)
sql_df.show()
```
```
+------------------+------+
|OperationSupported|  freq|
+------------------+------+
|        IN COUNTRY|546215|
+------------------+------+
```

In [85]:
```
# Q6 Which month saw the highest number of missions?
```

In [97]:
```
from pyspark.sql.functions import month
```

In [91]:
```
#missions_per_day = json_df.groupBy(json_df['ContryFlyingMission'],json_df['MissionDate']).count()
json_df.printSchema()
```
```
root
 |-- AirCraft: string (nullable = true)
 |-- ContryFlyingMission: string (nullable = true)
 |-- MissionDate: string (nullable = true)
 |-- OperationSupported: string (nullable = true)
 |-- PeriodOfDay: string (nullable = true)
 |-- TakeoffLocation: string (nullable = true)
 |-- TargetCountry: string (nullable = true)
 |-- TimeOnTarget: double (nullable = true)
 |-- WeaponType: string (nullable = true)
 |-- WeaponsLoadedWeight: long (nullable = true)
```

In [92]:
```
json_df = json_df.withColumn("MissionDate", col("MissionDate").cast("date"))
```

In [98]:
```
json_df = json_df.withColumn("Month", month("MissionDate"))
```

In [103…
```
json_df.groupBy(json_df['Month']).count().orderBy("count", ascending=False).head()
```

Out[103…
```
Row(Month=5, count=457431)
```

In [104…
```
# Q7 What was the most used aircraft type during the war (in terms of number of missions)?
```

In [107…
```
path_aircraft = '/content/drive/MyDrive/IDMP Assignment 4 files/Aircraft_Glossary.json'
aircraft_df = spark.read.json(path_aircraft)
aircraft_df.createOrReplaceTempView('aircrafts')
```

In [125…
```
query = """ select AirCraftType, count(MissionDate) as freq from
            operations o join aircrafts a on
            o.Aircraft = a.Aircraft
            group by AirCraftType
            order by freq desc
            limit 1
            """
```

In [126…
```
sql_df = spark.sql(query)
sql_df.show()
```
```
+------------------+-------+
|      AirCraftType|   freq|
+------------------+-------+
|Fighter Jet Bomber|1073126|
+------------------+-------+
```

In [ ]:
```

```