Adv. Database Management System

HW-01

1. **[25 pt]** Consider the following table in an operational astronomical DB:
   Star *(star_name, star_age, star_diameter, star_mass, star_temperature)*
   Assume that *star_age, star_diameter, star_mass, star_temperature* are categories, not numbers (e.g., for *star_age* values could be *'old', 'young',* etc.*).* Consider also a DW fact table ***StarFact***describing one measure (*total_number_of_stars*) and four dimensions (*age, diameter, mass,* and *temperature*).

   (a) *[5 pt]* Specify an SQL routing to load the Data Warehouse ***StarFact*** table from the operational ***Star*** table.
   (b) (b) *[10 pt]* Specify all views that you could generate from the ***StarFact*** table using CUBE operator (provide names of the views and corresponding group by queries you need to generate those views).
   (c) (c) *[10 pt]* Give an example of two OLAP queries that could be executed using two different views that you generated.

   Solution:

   a) INSERT INTO StarFact (total_number_of_stars, age, diameter, mass, temperature)
   SELECT COUNT(*), star_age, star_diameter, star_mass, star_temperature
   FROM Star
   GROUP BY star_age, star_diameter, star_mass, star_temperature;

   b) StarFact_Cube_Age
   CREATE VIEW StarFact_Cube_Age AS
   SELECT age, SUM(total_number_of_stars) AS total_stars
   FROM StarFact
   GROUP BY age WITH CUBE;

   StarFact_Cube_Diameter
   CREATE VIEW StarFact_Cube_Diameter AS
   SELECT diameter, SUM(total_number_of_stars) AS total_stars
   FROM StarFact
   GROUP BY diameter WITH CUBE;

   StarFact_Cube_ Mass
   CREATE VIEW StarFact_Cube_Mass AS
   SELECT mass, SUM(total_number_of_stars) AS total_stars
   FROM StarFact
   GROUP BY mass WITH CUBE;

StarFact_Cube_Temperature
CREATE VIEW StarFact_Cube_Temperature AS
SELECT temperature, SUM(total_number_of_stars) AS total_stars
FROM StarFact
GROUP BY temperature WITH CUBE;

c) **OLAP Query 1 using StarFact_Cube_Age**:
SELECT age, SUM(total_stars) AS total_stars
FROM StarFact_Cube_Age
GROUP BY age;

OLAP Query 2 using StarFact_Cube_Diameter:
SELECT diameter, SUM(total_stars) AS total_stars
FROM StarFact_Cube_Diameter
GROUP BY diameter;

2. Consider the following Data Warehouse tables:
Specify in SQL and show the results of the following OLAP queries:
a)s[3 pt] Find total money spent by each customer.
b) [3 pt] Find total sale amount for each store. Specify an SQL expression to perform the following roll up query:
c) [4 pt] Given total price by customers, get total price by gender. Show the result of this query.
Specify a linear algebra expression to perform the following drill down query:
d) [3 pt] Given sales by state, get sales by store.
e) [2pt] Show the result of the drilling down using Least Squares Method
f) [5pt] Show a linear algebra expression and the result of the drilling down using an assumption that total sale of all the stores in PA is the same as all the stores in NY.
g) [5pt] Show a linear algebra expression and the result of the drilling down using an assumption the data is smooth.
h) [5pt] Compute RMSE to for cases (e), (f) and (g).

Solution:

a) SELECT Customer.Cid, Customer.Name, SUM(Transaction.Total_sale) AS Total_Money_Spent
FROM Customer
JOIN Transaction ON Customer.Cid = Transaction.Cid
GROUP BY Customer.Cid, Customer.Name;

b) SELECT Store.Sid, Store.S_name, SUM(Transaction.Total_sale) AS Total_Sale_Amount
FROM Store
JOIN Transaction ON Store.Sid = Transaction.Sid
GROUP BY Store.Sid, Store.S_name;

c) SELECT

```
SELECT
  CASE
    WHEN Customer.Gender IS NULL THEN 'Total Price'
    ELSE Customer.Gender
  END AS Gender,
  SUM(Transaction.Total_sale) AS Total_Price
FROM
  Transaction
JOIN
  Customer ON Transaction.Cid = Customer.Cid
GROUP BY
  Customer.Gender WITH ROLLUP;
```

d)

```
x1 = 30
x2 = 30
x3 = 25
x4 = 90
PA = x1+x2
NYC = x3+x4


A = [1 1 0 0; 0 0 1 1];



y = [60;115];
x = pinv(A)*y;
x_gt = [30 30 25 90]';
rmse = sqrt(mean((x-x_gt).^2));
disp(rmse)
```

```
x1 = 30
x2 = 30
x3 = 25
x4 = 90
PA = 60
NYC = 115

22.9810
```

e)

```
y = [60;115];
A = [1 1 0 0; 0 0 1 1];
x = A \ y;
disp(x)
```

```
 60
  0
115
  0
```

f)

```
x1 = 50
x2 = 50
x3 = 25
x4 = 75
PA = x1+x2
NYC = x3+x4


A = [1 1 0 0; 0 0 1 1;0 0 1 -1];




y = [100;100;0];
x = pinv(A)*y;
x_gt = [50 50 25 75]';
rmse = sqrt(mean((x-x_gt).^2));
disp(rmse)
```

```
x1 = 50
x2 = 50
x3 = 25
x4 = 75
PA = 100
NYC = 100

17.6777
```

g)

```
y = [60;115;0];
A = [1 1 0 0; 0 0 1 1; 0 0 1 -1];
[Asm, ysm] = sm_constraints(A, y)
x = pinv(Asm)*ysm;
disp(x)
```

```
Asm = 6x4
     1     1     0     0
     0     0     1     1
     0     0     1    -1
    -1     0     0     0
     0    -1     0     0
     0     0     0     0

ysm = 6x1
    60
   115
     0
     0
     0
     0

   20.0000
   20.0000
   57.5000
   57.5000
```

h)

(e)

```
y = [60;115];
A = [1 1 0 0; 0 0 1 1];
x = A \ y;
disp(x)

x_gt = [30 30 25 90]';
rmse = sqrt(mean((x-x_gt).^2));
disp(rmse)
```

```
60
0
115
0

67.0820
```

(f)

```
y = [60;115;0];
A = [1 1 0 0; 0 0 1 1; 0 0 1 -1];
xgt = [30 30 25 90]
rmse = sqrt(mean((x-xgt).^2));
disp(rmse)
```

```
xgt = 1x4
    30    30    25    90

20.6912   20.6912   23.2513   54.5722
```

(g)

```
y = [60;115;0];
A = [1 1 0 0; 0 0 1 1; 0 0 1 -1];
[Asm, ysm] = sm_constraints(A, y)
x = pinv(Asm)*ysm;
x_gt = [30 30 25 90]
RMSE = sqrt(mean((x-x_gt).^2));
disp(RMSE)
```

```
Asm = 6x4
     1     1     0     0
     0     0     1     1
     0     0     1    -1
    -1     0     0     0
     0    -1     0     0
     0     0     0     0

ysm = 6x1
    60
   115
     0
     0
     0
     0

x_gt = 1x4
    30    30    25    90

20.6912   20.6912   23.2513   54.5722
```

3. **[25 pts]** MongoDB: Use the data (**primer-dataset.json**) to finish the following questions, show the query as well as the screenshot of result:
a) Select the data with name contains “Ham” at the beginning, output the second 5 restaurant id by descending order.
b) Count the number of restaurants whose name contains “Ham” and grade score greater than or equal to 20.
c) Use the text search function in the lab slides 27 of ‘Lab_NoSQL.ppt’, search the restaurant contain text “American” but exclude “Burger”, output the first 1 restaurant with the text score.
d) Refer to lab slides 28 and 29 of ‘Lab_NoSQL.ppt’, List the first place in the circle of the location (-74, 40) with radius 12.

Solution:

a)
```
db.collection.find(
   { "name": { "$regex": /^Ham/i } },
   { "restaurant_id": 1, "_id": 0 }
).sort(
   { "restaurant_id": -1 }
).skip(1).limit(5);
```

```
< {
    restaurant_id: '50016357'
  }
  {
    restaurant_id: '50012471'
  }
  {
    restaurant_id: '50011706'
  }
  {
    restaurant_id: '50005615'
  }
  {
    restaurant_id: '50001242'
  }
```

b) db.mycollection.count({
"name": { "$regex": /Ham/i },
"grades": { "$elemMatch": { "score": { "$gte": 20 } } } });

```
> db.mycollection.count({
      "name": { "$regex": /Ham/i },
      "grades": { "$elemMatch": { "score": { "$gte": 20 } } }
  });
< 32
```

c) db.mycollection.find({$text:{search: “American -Burger”}}, {score:{$meta:”textscore}}).limit(1).pretty();

```
> db.mycollection.find({ $text: { $search: "American -Burger" } }, { score: { $meta: "textScore" } }).limit(1).pretty();
< {
    _id: ObjectId('65d5381ea4a3670b2b567c79'),
    address: {
      building: '4',
      coord: [
        -74.01321329999999,
        40.7013064
      ],
      street: 'South Street',
      zipcode: '10004'
    },
    borough: 'Manhattan',
    cuisine: 'American',
```

```
    grades: [
      {
        date: 2014-09-18T00:00:00.000Z,
        grade: 'C',
        score: 28
      },
      {
        date: 2013-08-17T00:00:00.000Z,
        grade: 'A',
        score: 13
      },
      {
        date: 2013-04-25T00:00:00.000Z,
        grade: 'B',
        score: 20
      },
      {
        date: 2012-03-22T00:00:00.000Z,
```

```
  {
    date: 2013-04-25T00:00:00.000Z,
    grade: 'B',
    score: 20
  },
  {
    date: 2012-03-22T00:00:00.000Z,
    grade: 'A',
    score: 9
  }
 ],
 name: 'Centra`L Market All American Grill ( Staten Island Ferry Terminal)',
 restaurant_id: '41586885',
 score: 0.5555555555555556
}
```

d) db.mycollection.createIndex({ "location": "2dsphere" })
db.mycollection.find({ "address.coord": { $geoWithin: { $centerSphere: [[-74, 40], 5] } } }).limit(1);

```
< {
    _id: ObjectId('65d53819a4a3670b2b564729'),
    address: {
      building: '1007',
      coord: [
        -73.856077,
        40.848447
      ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'Bronx',
    cuisine: 'Bakery',
    grades: [
      {
        date: 2014-03-03T00:00:00.000Z,
        grade: 'A',
        score: 2
```

```
        score: 2
      },
      {
        date: 2013-09-11T00:00:00.000Z,
        grade: 'A',
        score: 6
      },
      {
        date: 2013-01-24T00:00:00.000Z,
        grade: 'A',
        score: 10
      },
      {
        date: 2011-11-23T00:00:00.000Z,
        grade: 'A',
        score: 9
      },
```

```
    },
    {
      date: 2011-11-23T00:00:00.000Z,
      grade: 'A',
      score: 9
    },
    {
      date: 2011-03-10T00:00:00.000Z,
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
}
```

e) db.createCollection("authors")
// Insert documents into the collection
db.authors.insertMany([
 {"author": "Dave", "score": 50, "views": 100},
 {"author": "Dave", "score": 80, "views": 1100},
 {"author": "Peter", "score": 80, "views": 120},
 {"author": "John", "score": 70, "views": 500},
 {"author": "Jess", "score": 60, "views": 984},
 {"author": "John", "score": 75, "views": 321},
 {"author": "Peter", "score": 85, "views": 143}
])
db.authors.aggregate([{ $match: { views: { $gte: 300, $lte: 1000 } } }, { $group: { _id: "$author", totalViews: { $sum: "$views" } } }]).forEach(function(doc) {printjson(doc);});

```
> db.createCollection("authors")
< { ok: 1 }
> db.authors.insertMany([
   {"author": "Dave", "score": 50, "views": 100},
   {"author": "Dave", "score": 80, "views": 1100},
```

```
    {"author": "Dave", "score": 50, "views": 100},
    {"author": "Dave", "score": 80, "views": 1100},
    {"author": "Peter", "score": 80, "views": 120},
    {"author": "John", "score": 70, "views": 500},
    {"author": "Jess", "score": 60, "views": 984},
    {"author": "John", "score": 75, "views": 321},
    {"author": "Peter", "score": 85, "views": 143}
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('65d54cef7a9c98e6fa656cf0'),
      '1': ObjectId('65d54cef7a9c98e6fa656cf1'),
      '2': ObjectId('65d54cef7a9c98e6fa656cf2'),
      '3': ObjectId('65d54cef7a9c98e6fa656cf3'),
      '4': ObjectId('65d54cef7a9c98e6fa656cf4'),
      '5': ObjectId('65d54cef7a9c98e6fa656cf5'),
      '6': ObjectId('65d54cef7a9c98e6fa656cf6')
    }
```

```
 });
< { _id: 'Jess', totalViews: 984 }
< { _id: 'John', totalViews: 821 }
```

4. **[20 pts]** Neo4j: Use '**producer_neo.py**' to import data into your No4j database, and then answer the following questions by modifying '**consumer_neo.py**', please provide the **consumer_neo.py** and result screenshot in your final submission.
   (a) Find the names of all people whose age is between 20 and 24 (include).
   (b) Find the count of students in the current data.
   (c) List the name of all people in the data and replace the "a" in the output string with "A".
   (d) Please find all the friend of Kary. Return related node(s).

   <u>Solution:</u>

```
(a) Names of all persons aged between 20 and 24 (inclusive): ['Kelly', 'alia']
(b) Count of students in the current data: 0
(c) Names of all people with 'a' replaced with 'A': ['Peter', 'Kelly', 'KAry', 'AliA']
(d) Friends of Kary: ['Peter']
```