QUERIES and OUPUT

MYSQL QUERIES and OUTPUT

1. Find agents who receive commissions between 0.12 and 0.14 (begin and end values are included). Return agent_code, name, city, and commission.

```
mysql> SELECT AGENT_CODE, AGENT_NAME, WORKING_AREA, COMMISSION
    -> FROM AGENTS
    -> WHERE COMMISSION BETWEEN 0.12 AND 0.14;
+------------+------------+-------------+------------+
| AGENT_CODE | AGENT_NAME | WORKING_AREA | COMMISSION |
+------------+------------+-------------+------------+
| A001       | Subbarao   | Bangalore    |       0.14 |
| A003       | Alex       | London       |       0.13 |
| A005       | Anderson   | Brisban      |       0.13 |
| A008       | Alford     | New York     |       0.12 |
| A010       | Santakumar | Chennai      |       0.14 |
| A012       | Lucida     | San Jose     |       0.12 |
+------------+------------+-------------+------------+
6 rows in set (0.02 sec)
```

2. Retrieve the details of the agent whose names begin with any letter between 'A' and 'L' (not inclusive). Return agent_code, name, city, commission.

```
mysql> SELECT AGENT_CODE, AGENT_NAME, WORKING_AREA, COMMISSION
    -> FROM AGENTS
    -> WHERE AGENT_NAME > 'A' AND AGENT_NAME < 'L';
+------------+------------+-------------+------------+
| AGENT_CODE | AGENT_NAME | WORKING_AREA | COMMISSION |
+------------+------------+-------------+------------+
| A003       | Alex       | London       |       0.13 |
| A004       | Ivan       | Torento      |       0.15 |
| A005       | Anderson   | Brisban      |       0.13 |
| A008       | Alford     | New York     |       0.12 |
| A009       | Benjamin   | Hampshair    |       0.11 |
+------------+------------+-------------+------------+
5 rows in set (0.00 sec)
```

3. Find all those customers who do.es not have any grade. Return customer_id, cust_name, city, grade, agent_code.

```
mysql> SELECT CUST_CODE, CUST_NAME, CUST_CITY, GRADE, AGENT_CODE
    -> FROM CUSTOMER
    -> WHERE GRADE IS NULL;
+-----------+-----------+-----------+-------+------------+
| CUST_CODE | CUST_NAME | CUST_CITY | GRADE | AGENT_CODE |
+-----------+-----------+-----------+-------+------------+
| C00026    | John Doe  | New York  |  NULL | A003       |
+-----------+-----------+-----------+-------+------------+
1 row in set (0.01 sec)
```

4. Find the highest purchase amount ordered by each customer. Return CUST_CODE, maximum purchase amount.

```
mysql> SELECT CUST_CODE, MAX(ORD_AMOUNT) AS maximum_purchase_amount
    -> FROM ORDERS
    -> GROUP BY CUST_CODE;
+-----------+------------------------+
| CUST_CODE | maximum_purchase_amount |
+-----------+------------------------+
| C00013    |                5500.00 |
| C00001    |                7000.00 |
| C00020    |                7000.00 |
| C00025    |                8000.00 |
| C00024    |                5500.00 |
| C00015    |                6500.00 |
| C00002    |                7500.00 |
| C00018    |                9000.00 |
| C00009    |                7200.00 |
| C00010    |                6800.00 |
+-----------+------------------------+
10 rows in set (0.01 sec)
```

5. Calculate total purchase amount of all orders. Return total purchase amount.

```
mysql> SELECT SUM(ORD_AMOUNT) AS total_purchase_amount
    -> FROM ORDERS;
+----------------------+
| total_purchase_amount |
+----------------------+
|              81000.00 |
+----------------------+
1 row in set (0.00 sec)
```

6. Determine the number of customers who received at least one grade for their activity.

```
mysql> SELECT COUNT(DISTINCT CUST_CODE) AS num_customers_with_grade
    -> FROM CUSTOMER
    -> WHERE GRADE IS NOT NULL;
+--------------------------+
| num_customers_with_grade |
+--------------------------+
|                       25 |
+--------------------------+
1 row in set (0.00 sec)
```

7. Find the highest purchase amount ordered by each customer on a particular date. Return, order date and highest purchase amount.

```
mysql> SELECT ORD_DATE, CUST_CODE, MAX(ORD_AMOUNT) AS highest_purchase_amount
    -> FROM ORDERS
    -> GROUP BY ORD_DATE, CUST_CODE;
+------------+-----------+------------------------+
| ORD_DATE   | CUST_CODE | highest_purchase_amount |
+------------+-----------+------------------------+
| 2023-09-21 | C00013    |                5500.00 |
| 2023-09-22 | C00001    |                7000.00 |
| 2012-10-10 | C00020    |                6000.00 |
| 2012-09-10 | C00025    |                8000.00 |
| 2023-09-25 | C00024    |                5500.00 |
| 2023-09-26 | C00015    |                6500.00 |
| 2023-09-27 | C00002    |                7500.00 |
| 2023-09-28 | C00018    |                9000.00 |
| 2023-09-28 | C00009    |                7200.00 |
| 2023-09-29 | C00010    |                6800.00 |
| 2023-09-21 | C00020    |                7000.00 |
+------------+-----------+------------------------+
11 rows in set (0.00 sec)
```

8. Find the highest order (purchase) amount by each customer on a particular order date. Filter the result by highest order (purchase) amount above 2000.00. Return CUST_CODE, order date and maximum purchase amount.

```
mysql> SELECT CUST_CODE, ORD_DATE, MAX(ORD_AMOUNT) AS max_purchase_amount
    -> FROM ORDERS
    -> GROUP BY CUST_CODE, ORD_DATE
    -> HAVING MAX(ORD_AMOUNT) > 2000.00;
+-----------+------------+--------------------+
| CUST_CODE | ORD_DATE   | max_purchase_amount |
+-----------+------------+--------------------+
| C00013    | 2023-09-21 |            5500.00 |
| C00001    | 2023-09-22 |            7000.00 |
| C00020    | 2012-10-10 |            6000.00 |
| C00025    | 2012-09-10 |            8000.00 |
| C00024    | 2023-09-25 |            5500.00 |
| C00015    | 2023-09-26 |            6500.00 |
| C00002    | 2023-09-27 |            7500.00 |
| C00018    | 2023-09-28 |            9000.00 |
| C00009    | 2023-09-28 |            7200.00 |
| C00010    | 2023-09-29 |            6800.00 |
| C00020    | 2023-09-21 |            7000.00 |
+-----------+------------+--------------------+
11 rows in set (0.00 sec)
```

9. Count all the orders generated on '2012-08-17'. Return number of orders.

```
mysql> SELECT COUNT(*) AS number_of_orders
    -> FROM ORDERS
    -> WHERE ORD_DATE = '2012-08-17';
+------------------+
| number_of_orders |
+------------------+
|                0 |
+------------------+
1 row in set (0.00 sec)
```

10. Find those agents who generated orders for their customers but are not located in the same city. Return ORD_NUM, cust_name, cust_code (orders table), agent_code (orders table).

```
mysql> SELECT ORD_NUM, (
    -> SELECT CUST_NAME FROM CUSTOMER WHERE CUST_CODE = ORDERS.CUST_CODE)
    -> AS cust_name, CUST_CODE, AGENT_CODE
    -> FROM ORDERS
    -> WHERE AGENT_CODE IN
    -> (SELECT AGENT_CODE FROM AGENTS WHERE WORKING_AREA !=
    -> (SELECT WORKING_AREA FROM CUSTOMER WHERE CUST_CODE = ORDERS.CUST_CODE));
+---------+-----------+-----------+------------+
| ORD_NUM | cust_name | CUST_CODE | AGENT_CODE |
+---------+-----------+-----------+------------+
|    1001 | Holmes    | C00013    | A007       |
|    1002 | Micheal   | C00001    | A003       |
|    1009 | Ramesh    | C00009    | A012       |
+---------+-----------+-----------+------------+
3 rows in set (0.00 sec)
```

11. Find those customers who are served by a salesperson and the salesperson earns commission in the range of 12% to 14% (Begin and end values are included.). Return cust_name AS "Customer", city AS "City".

```
mysql> SELECT
    ->   ORD_NUM, (
    ->     SELECT CUST_NAME FROM CUSTOMER
    ->       WHERE CUST_CODE = ORDERS.CUST_CODE )
    ->           AS cust_name, CUST_CODE, AGENT_CODE
    ->         FROM ORDERS
    ->       WHERE AGENT_CODE IN (
    ->     SELECT AGENT_CODE
    ->         FROM AGENTS
    ->       WHERE WORKING_AREA != (
    ->     SELECT WORKING_AREA
    ->         FROM CUSTOMER
    ->       WHERE CUST_CODE = ORDERS.CUST_CODE
    ->       )
    ->   );
+---------+-----------+-----------+------------+
| ORD_NUM | cust_name | CUST_CODE | AGENT_CODE |
+---------+-----------+-----------+------------+
|    1001 | Holmes    | C00013    | A007       |
|    1002 | Micheal   | C00001    | A003       |
|    1009 | Ramesh    | C00009    | A012       |
+---------+-----------+-----------+------------+
3 rows in set (0.00 sec)
```

12. Find all orders executed by the salesperson and ordered by the customer whose grade is greater than or equal to 200. Compute purch_amt*commission as “Commission”. Return customer name, commission as “Commission%” and Commission.

```
mysql> SELECT
    ->     C.CUST_NAME AS "Customer Name",
    ->     (O.ORD_AMOUNT * A.COMMISSION) AS "Commission",
    ->     (A.COMMISSION * 100) AS "Commission%"
    -> FROM
    ->     ORDERS O, AGENTS A, CUSTOMER C
    -> WHERE
    ->     O.AGENT_CODE = A.AGENT_CODE
    ->     AND O.CUST_CODE = C.CUST_CODE
    ->     AND C.GRADE >= 200;
+---------------+------------+-------------+
| Customer Name | Commission | Commission% |
+---------------+------------+-------------+
| Ravindran     |  1200.0000 |       15.00 |
+---------------+------------+-------------+
1 row in set (0.00 sec)
```

13. Find the order values greater than the average order value of 10th October 2012. Return ord_no, purch_amt, ord_date, cust_code, agent_code.

```
mysql> SELECT ORD_NUM, ORD_AMOUNT AS "purch_amt",
    ->     ORD_DATE, CUST_CODE, AGENT_CODE
    ->     FROM ORDERS
    ->     WHERE ORD_AMOUNT > (
    ->     SELECT AVG(ORD_AMOUNT) FROM ORDERS
    ->     WHERE ORD_DATE = '2012-10-10'
    -> );
+---------+-----------+------------+-----------+------------+
| ORD_NUM | purch_amt | ORD_DATE   | CUST_CODE | AGENT_CODE |
+---------+-----------+------------+-----------+------------+
|    1002 |   7000.00 | 2023-09-22 | C00001    | A003       |
|    1004 |   8000.00 | 2012-09-10 | C00025    | A011       |
|    1006 |   6500.00 | 2023-09-26 | C00015    | A003       |
|    1007 |   7500.00 | 2023-09-27 | C00002    | A008       |
|    1008 |   9000.00 | 2023-09-28 | C00018    | A005       |
|    1009 |   7200.00 | 2023-09-28 | C00009    | A012       |
|    1010 |   6800.00 | 2023-09-29 | C00010    | A009       |
|   10012 |   7000.00 | 2023-09-21 | C00020    | A008       |
+---------+-----------+------------+-----------+------------+
8 rows in set (0.00 sec)
```

14. Find the sums of the amounts from the orders table, grouped by date, and eliminate all dates where the sum was not at least 1000.00 above the maximum order amount for that date.

```
mysql> SELECT
    ->     ORD_DATE,
    ->     SUM(ORD_AMOUNT) AS TotalAmount
    -> FROM
    ->     ORDERS
    -> GROUP BY
    ->     ORD_DATE
    -> HAVING
    ->     SUM(ORD_AMOUNT) >= (MAX(ORD_AMOUNT) + 1000.00);
+------------+-------------+
| ORD_DATE   | TotalAmount |
+------------+-------------+
| 2023-09-21 |    17500.00 |
| 2023-09-28 |    16200.00 |
+------------+-------------+
2 rows in set (0.00 sec)
```

15. find details of all orders excluding those with ord_date equal to '2012-09-10' and agent_code higher than 5005 or purch_amt greater than 1000.Return ord_no, purch_amt, ord_date, cust_code and salesman_id.

```
mysql> SELECT
    ->     ORD_NUM AS "ord_no",
    ->     ORD_AMOUNT AS "purch_amt",
    ->     ORD_DATE,
    ->     CUST_CODE,
    ->     AGENT_CODE AS "salesman_id"
    -> FROM
    ->     ORDERS
    -> WHERE
    ->     ORD_DATE != '2012-09-10'
    ->     AND (AGENT_CODE <= 'A005' OR ORD_AMOUNT <= 1000.00);
+--------+-----------+------------+-----------+-------------+
| ord_no | purch_amt | ORD_DATE   | CUST_CODE | salesman_id |
+--------+-----------+------------+-----------+-------------+
|   1002 |   7000.00 | 2023-09-22 | C00001    | A003        |
|   1006 |   6500.00 | 2023-09-26 | C00015    | A003        |
|   1008 |   9000.00 | 2023-09-28 | C00018    | A005        |
|  10011 |   5500.00 | 2023-09-21 | C00013    | A003        |
+--------+-----------+------------+-----------+-------------+
4 rows in set (0.00 sec)
```

MONGODB QUERIES and OUTPUT

1. Select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

```
dm> db.address.find(
...   { "grades.score": { $mod: [7, 0] } },
...   { restaurant_id: 1, name: 1, grades: 1 }
... );
[
  {
    _id: ObjectId("650b4208f68f9c79aed08840"),
    grades: [
      { date: { '$date': 1393804800000 }, grade: 'A', score: 2 },
      { date: { '$date': 1378857600000 }, grade: 'A', score: 6 },
      { date: { '$date': 1358985600000 }, grade: 'A', score: 10 },
      { date: { '$date': 1322006400000 }, grade: 'A', score: 9 },
      { date: { '$date': 1299715200000 }, grade: 'B', score: 14 }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  },
  {
    _id: ObjectId("650b42d6f68f9c79aed08845"),
    grades: [
      { date: { '$date': 1402358400000 }, grade: 'A', score: 5 },
      { date: { '$date': 1370390400000 }, grade: 'A', score: 7 },
      { date: { '$date': 1334275200000 }, grade: 'A', score: 12 },
      { date: { '$date': 1318377600000 }, grade: 'A', score: 12 }
    ],
    name: 'Riviera Caterer',
    restaurant_id: '40356018'
  },
```

2. Find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name

```
dm> db.address.find( { borough: { $regex: /mon/i } }, { name: 1, borough: 1, "address.coord": 1, cuisine: 1 } )
[
  {
    _id: ObjectId("650b4208f68f9c79aed08840"),
    address: { coord: [ -73.856077, 40.848447 ] },
    borough: 'comon',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop'
  }
]
```

3. Find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan.

```
dm> db.address.find({
...   borough: "Manhattan",
...   grades: {
...     $elemMatch: { score: { $lt: 5 } }
...   }
... })
[
  {
    _id: ObjectId("650b42bef68f9c79aed08844"),
    address: {
      building: '351',
      coord: [ -73.98513559999999, 40.7676919 ],
      street: 'West   57 Street',
      zipcode: '10019'
    },
    borough: 'Manhattan',
    cuisine: 'Irish',
    grades: [
      { date: { '$date': 1409961600000 }, grade: 'A', score: 2 },
      { date: { '$date': 1374451200000 }, grade: 'A', score: 11 },
      { date: { '$date': 1343692800000 }, grade: 'A', score: 12 },
      { date: { '$date': 1325116800000 }, grade: 'A', score: 12 }
    ],
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841'
  },
  {
    _id: ObjectId("650b45fef68f9c79aed08853"),
    address: {
      building: '1',
      coord: [ -73.96926909999999, 40.7685235 ],
      street: 'East   66 Street',
      zipcode: '10065'
    },
    borough: 'Manhattan',
    cuisine: 'American ',
    grades: [
      { date: { '$date': 1399420800000 }, grade: 'A', score: 3 },
      { date: { '$date': 1367539200000 }, grade: 'A', score: 4 },
      { date: { '$date': 1335744000000 }, grade: 'A', score: 6 },
      { date: { '$date': 1324944000000 }, grade: 'A', score: 0 }
    ],
    name: '1 East 66Th Street Kitchen',
    restaurant_id: '40359480'
  }
```

4. Find the restaurants that have at least one grade with a score of less than 5 and that are located in the borough of Manhattan or Brooklyn, and their cuisine is not American or Chinese.

```
dm> db.address.find({
...   $or: [
...     { borough: "Manhattan" },
...     { borough: "Brooklyn" }
...   ],
...   $or: [
...     { "grades.score": 2 },
...     { "grades.score": 6 }
...   ],
...   cuisine: { $ne: "American" }
... })
[
  {
    _id: ObjectId("650b4208f68f9c79aed08840"),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'comon',
    cuisine: 'Bakery',
    grades: [
      { date: { '$date': 1393804800000 }, grade: 'A', score: 2 },
      { date: { '$date': 1378857600000 }, grade: 'A', score: 6 },
      { date: { '$date': 1358985600000 }, grade: 'A', score: 10 },
      { date: { '$date': 1322006400000 }, grade: 'A', score: 9 },
      { date: { '$date': 1299715200000 }, grade: 'B', score: 14 }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  },
  {
    _id: ObjectId("650b42bef68f9c79aed08844"),
    address: {
      building: '351',
      coord: [ -73.98513559999999, 40.7676919 ],
      street: 'West   57 Street',
      zipcode: '10019'
    },
    borough: 'Manhattan',
    cuisine: 'Irish',
    grades: [
      { date: { '$date': 1409961600000 }, grade: 'A', score: 2 },
      { date: { '$date': 1374451200000 }, grade: 'A', score: 11 },
      { date: { '$date': 1343692800000 }, grade: 'A', score: 12 },
      { date: { '$date': 1325116800000 }, grade: 'A', score: 12 }
    ],
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841'
  },
```

5. Find the restaurants that have a grade with a score of 2 or a grade with a score of 6 and are located in the borough of Manhattan or Brooklyn, and their cuisine is not American.

```
dm> db.address.find({
...   $or: [
...     { borough: "Manhattan" },
...     { borough: "Brooklyn" }
...   ],
...   $or: [
...     { "grades.score": 2 },
...     { "grades.score": 6 }
...   ],
...   cuisine: { $ne: "American" }
... })
[
  {
    _id: ObjectId("650b4208f68f9c79aed08840"),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462'
    },
    borough: 'comon',
    cuisine: 'Bakery',
    grades: [
      { date: { '$date': 1393804800000 }, grade: 'A', score: 2 },
      { date: { '$date': 1378857600000 }, grade: 'A', score: 6 },
      { date: { '$date': 1358985600000 }, grade: 'A', score: 10 },
      { date: { '$date': 1322006400000 }, grade: 'A', score: 9 },
      { date: { '$date': 1299715200000 }, grade: 'B', score: 14 }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  },
  {
    _id: ObjectId("650b42bef68f9c79aed08844"),
    address: {
      building: '351',
      coord: [ -73.98513559999999, 40.7676919 ],
      street: 'West   57 Street',
      zipcode: '10019'
    },
    borough: 'Manhattan',
    cuisine: 'Irish',
    grades: [
      { date: { '$date': 1409961600000 }, grade: 'A', score: 2 },
      { date: { '$date': 1374451200000 }, grade: 'A', score: 11 },
      { date: { '$date': 1343692800000 }, grade: 'A', score: 12 },
      { date: { '$date': 1325116800000 }, grade: 'A', score: 12 }
    ],
    name: 'Dj Reynolds Pub And Restaurant',
    restaurant_id: '30191841'
  },
```

6. Find the average score for each restaurant.

```
dm> db.address.aggregate([
...   {
...     $unwind: "$grades"
...   },
...   {
...     $group: {
...       _id: "$restaurant_id",
...       avgScore: { $avg: "$grades.score" }
...     }
...   },
...   {
...     $project: {
...       _id: 0,
...       restaurant_id: "$_id",
...       avgScore: 1
...     }
...   }
... ]);
[
  { avgScore: 27.6, restaurant_id: '40358429' },
  { avgScore: 9.25, restaurant_id: '40361708' },
  { avgScore: 6, restaurant_id: '40357217' },
  { avgScore: 17, restaurant_id: '40356151' },
  { avgScore: 3.5, restaurant_id: '40357437' },
  { avgScore: 10.666666666666666, restaurant_id: '40362098' },
  { avgScore: 7.5, restaurant_id: '40361998' },
  { avgScore: 9.25, restaurant_id: '30191841' },
  { avgScore: 18.6, restaurant_id: '40362264' },
  { avgScore: 5.666666666666667, restaurant_id: '40361390' },
  { avgScore: 3.25, restaurant_id: '40359480' },
  { avgScore: 26, restaurant_id: '' },
  { avgScore: 17.75, restaurant_id: '40356068' },
  { avgScore: 13.75, restaurant_id: '30112340' },
  { avgScore: 10, restaurant_id: '40356483' },
  { avgScore: 9.25, restaurant_id: '40361606' },
  { avgScore: 8.25, restaurant_id: '40356731' },
  { avgScore: 10.666666666666666, restaurant_id: '40361322' },
  { avgScore: 9, restaurant_id: '40356018' },
  { avgScore: 12.2, restaurant_id: '40361521' }
]
```

7. Find the count of restaurants for each cuisine.

```
dm> db.address.aggregate([
...   { $group: { _id: "$cuisine", count: { $sum: 1 } } }
... ])
[
  { _id: 'Irish', count: 1 },
  { _id: 'Chinese', count: 1 },
  { _id: 'American ', count: 13 },
  { _id: 'Chicken', count: 1 },
  { _id: 'Jewish/Kosher', count: 3 },
  { _id: 'Ice Cream, Gelato, Yogurt, Ices', count: 2 },
  { _id: 'Bakery', count: 1 },
  { _id: 'Delicatessen', count: 5 },
  { _id: 'Hamburgers', count: 1 }
]
dm>
```

8. Find the count of restaurants that received a grade of 'A' for each cuisine.

```
dm> db.address.aggregate([
...   { $unwind: "$grades" },
...   { $match: { "grades.grade": "A" } },
...   { $group: { _id: "$cuisine", count: { $sum: 1 } } }])
[
  { _id: 'Bakery', count: 4 },
  { _id: 'Chicken', count: 6 },
  { _id: 'American ', count: 37 },
  { _id: 'Jewish/Kosher', count: 10 },
  { _id: 'Delicatessen', count: 21 },
  { _id: 'Hamburgers', count: 3 },
  { _id: 'Ice Cream, Gelato, Yogurt, Ices', count: 7 },
  { _id: 'Chinese', count: 1 },
  { _id: 'Irish', count: 4 }
]
```

9. Find the number of restaurants that have been graded in each month of the year.

```
dm> db.address.aggregate([
...   { $unwind: "$grades" },
...   {
...     $group: {
...       _id: {
...         month: { $month: "$grades.date.date" },
...         year: { $year: "$grades.date.date" },
...       },
...       count: { $sum: 1 },
...     },
...   },
...   { $sort: { "_id.year": 1, "_id.month": 1 } },
... ]);
[
  { _id: { month: null, year: null }, count: 106 },
  { _id: { month: 1, year: 2021 }, count: 3 }
]
```

10. Find the name and address of the restaurants that received a grade of 'A' on a specific date.

```
dm> db.address.find(
...   {
...     grades: {
...       $elemMatch: {
...         date: { date: ISODate("2021-01-15T06:31:15.000Z") },
...         grade: "A",
...       },
...     },
...   },
...   { name: 1, "address.building": 1, "address.street": 1, _id: 0 }
... );
[
  {
    address: { building: '730', street: 'Columbus Avenue' },
    name: 'P & S Deli Grocery'
  },
  {
    address: { building: '730', street: 'Columbus Avenue' },
    name: 'P & S Deli Grocery'
  }
]
```