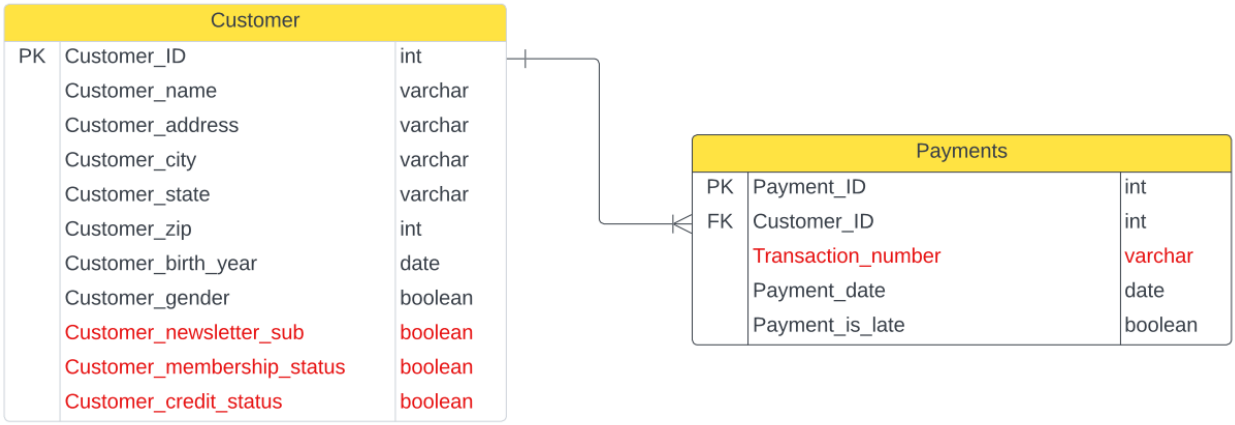
NoSQL Assignment

Fahad Ahmad

225812550

Q1. Design an ERD that represents the entities and relationships of the database that contains the data from those two files. (Note: some fields in the txt file have "no-explanation". What do you think they stand for? Give them "meaning" in your ERD…)



This is an ERD diagram representing the entities and relationship of our two files, one as customer and the other as payments. The attributes highlighted in red font are the fields that had no explanation so we have given them a name and meaning that can be relatable to the database. We have customer_id as primary key for customer table and payment_id for payment table as primary key. Both tables have been connected through the customer_id so customer_id becomes a foreign key in payments table. We have given one to many relationship between customer and payments because one customer can have multiple payments.

Q2. Store the records in a key-value format in Redis.

The following is the syntax to record the data in key-value format in Redis. I have attached along screen shots for redis recording for first 2 customers and payments to show how it is done and what is the output message. The syntax for all customers and payments are written below that can be just put into Redis and executed.

Customer data

SET customer:171 '{"Customer_name": "Madelyn Hensley", "Customer_address": "10075 Thierer Plaza", "Customer_city": "New York", "Customer_state": "New York", "Customer_zip": 81377,

"Customer_birth_year": 1976, "Customer_gender": "M", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": false}'

SET customer:172 '{"Customer_name": "Lonny Foster", "Customer_address": "23901 Park Meadow Dr", "Customer_city": "Austin", "Customer_state": "Texas", "Customer_zip": 13498, "Customer_birth_year": 1981, "Customer_gender": "F", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": false}'

SET customer:173 '{"Customer_name": "Karina Livingston", "Customer_address": "95 Anderson Park", "Customer_city": "Chattanooga", "Customer_state": "Tennessee", "Customer_zip": 94518, "Customer_birth_year": 1977, "Customer_gender": "F", "Customer_newsletter_sub": true, "Customer_membership_status": true, "Customer_credit_status": true}'

SET customer:174 '{"Customer_name": "Avery Mccormick", "Customer_address": "09992 Sunfield Parkway", "Customer_city": "Chicago", "Customer_state": "Illinois", "Customer_zip": 38300, "Customer_birth_year": 1992, "Customer_gender": "F", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": false}'

SET customer:175 '{"Customer_name": "Peter King", "Customer_address": "0486 Dryden Road", "Customer_city": "Chicago", "Customer_state": "Illinois", "Customer_zip": 21012, "Customer_birth_year": 1991, "Customer_gender": "M", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": true}'

SET customer:176 '{"Customer_name": "Bret Ibarra", "Customer_address": "14 Transport Place", "Customer_city": "San Diego", "Customer_state": "California", "Customer_zip": 12865, "Customer_birth_year": 1984, "Customer_gender": "M", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": false}'

SET customer:177 '{"Customer_name": "Leonardo Wheeler", "Customer_address": "806 Corry Crossing", "Customer_city": "New York", "Customer_state": "New York", "Customer_zip": 44464, "Customer_birth_year": 1972, "Customer_gender": "F", "Customer_newsletter_sub": false, "Customer_membership_status": false, "Customer_credit_status": true}'

SET customer:178 '{"Customer_name": "Bennett Noble", "Customer_address": "8 South Terrace", "Customer_city": "Hixson", "Customer_state": "Tennessee", "Customer_zip": 52890, "Customer_birth_year": 1989, "Customer_gender": "F", "Customer_newsletter_sub": false, "Customer_membership_status": false, "Customer_credit_status": true}'

SET customer:179 '{"Customer_name": "Marcia Mathews", "Customer_address": "0380 Knutson Road", "Customer_city": "Dallas", "Customer_state": "Texas", "Customer_zip": 80477, "Customer_birth_year": 1967, "Customer_gender": "F", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": true}'

SET customer:180 '{"Customer_name": "Avis Kramer", "Customer_address": "49624 Hanover Junction", "Customer_city": "New York", "Customer_state": "New York", "Customer_zip": 62542, "Customer_birth_year": 1965, "Customer_gender": "M", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": true}'

SET customer:181 '{"Customer_name": "Lynnette Tate", "Customer_address": "30741 Paget Court", "Customer_city": "New York", "Customer_state": "New York", "Customer_zip": 34886, "Customer_birth_year": 1987, "Customer_gender": "M", "Customer_newsletter_sub": true, "Customer_membership_status": false, "Customer_credit_status": true}'

SET customer:182 '{"Customer_name": "Lakisha Estrada", "Customer_address": "50 Dahle Crossing", "Customer_city": "Dallas", "Customer_state": "Texas", "Customer_zip": 16042, "Customer_birth_year": 1991, "Customer_gender": "F", "Customer_newsletter_sub": false, "Customer_membership_status": true, "Customer_credit_status": false}'

SET customer:183 '{"Customer_name": "Bill Silva", "Customer_address": "4 Mcbride Crossing", "Customer_city": "Detroit", "Customer_state": "Michigan", "Customer_zip": 15871, "Customer_birth_year": 1968, "Customer_gender": "M", "Customer_newsletter_sub": true, "Customer_membership_status": true, "Customer_credit_status": true}'

Payment data

SET payment:1 '{"Customer_id": 181, "Transaction_number": "146268743-8", "Payment_date": "23-8-2020", "Payment_is_late": true}'

SET payment:2 '{"Customer_id": 172, "Transaction_number": "396589804-7", "Payment_date": "28-9-2020", "Payment_is_late": false}'

SET payment:3 '{"Customer_id": 183, "Transaction_number": "553753031-8", "Payment_date": "25-9-2020", "Payment_is_late": false}'

SET payment:4 '{"Customer_id": 183, "Transaction_number": "559786593-4", "Payment_date": "13-12-2018", "Payment_is_late": true}'

SET payment:5 '{"Customer_id": 175, "Transaction_number": "108659198-9", "Payment_date": "13-9-2016", "Payment_is_late": true}'

SET payment:6 '{"Customer_id": 176, "Transaction_number": "360007723-2", "Payment_date": "27-9-2016", "Payment_is_late": false}'

SET payment:7 '{"Customer_id": 177, "Transaction_number": "238309554-X", "Payment_date": "28-11-2014", "Payment_is_late": false}'

SET payment:8 '{"Customer_id": 178, "Transaction_number": "694690715-8", "Payment_date": "31-5-2014", "Payment_is_late": true}'

SET payment:9 '{"Customer_id": 179, "Transaction_number": "318241713-5", "Payment_date": "15-7-2016", "Payment_is_late": true}'

SET payment:10 '{"Customer_id": 180, "Transaction_number": "84360172-2", "Payment_date": "07-06-2017", "Payment_is_late": false}'

SET payment:11 '{"Customer_id": 181, "Transaction_number": "807633856-8", "Payment_date": "07-05-2018", "Payment_is_late": false}'

SET payment:12 '{"Customer_id": 182, "Transaction_number": "845886260-4", "Payment_date": "23-4-2014", "Payment_is_late": false}'

SET payment:13 '{"Customer_id": 183, "Transaction_number": "270161074-X", "Payment_date": "19-7-2016", "Payment_is_late": true}'

SET payment:14 '{"Customer_id": 171, "Transaction_number": "887428332-0", "Payment_date": "07-04-2020", "Payment_is_late": false}'

SET payment:15 '{"Customer_id": 172, "Transaction_number": "401129380-4", "Payment_date": "23-8-2015", "Payment_is_late": true}'

SET payment:16 '{"Customer_id": 173, "Transaction_number": "277406266-7", "Payment_date": "27-12-2019", "Payment_is_late": false}'

SET payment:17 '{"Customer_id": 174, "Transaction_number": "851112155-2", "Payment_date": "12-04-2019", "Payment_is_late": true}'

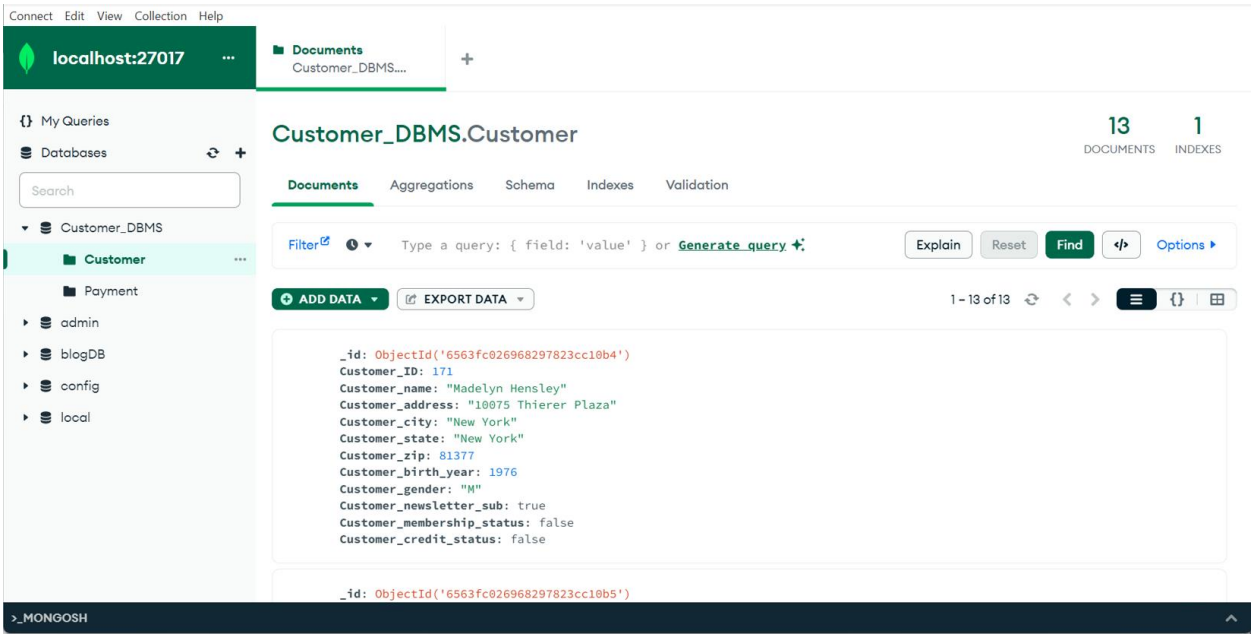
```
redis> SET customer:171 '{"Customer_name": "Madelyn
Hensley", "Customer_address": "10075 Thierer Plaza",
"Customer_city": "New York", "Customer_state": "New
York", "Customer_zip": 81377, "Customer_birth_year":
1976, "Customer_gender": "M", "Customer_newsletter_sub":
true, "Customer_membership_status": false,
"Customer_credit_status": false}'
"OK"
redis> SET payment:14 '{"Customer_id": 171,
"Transaction_number": "887428332-0", "Payment_date":
"07-04-2020", "Payment_is_late": false}'
"OK"
redis>
```

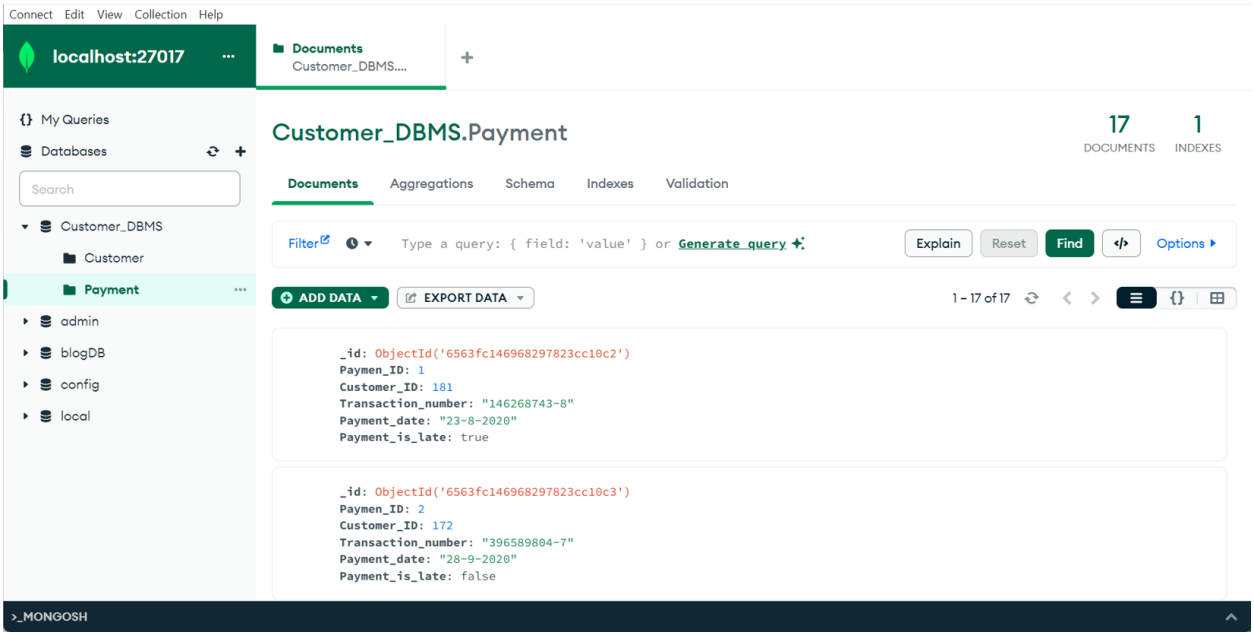
```
redis> SET customer:172 '{"Customer_name": "Lonny
Foster", "Customer_address": "23901 Park Meadow Dr",
"Customer_city": "Austin", "Customer_state": "Texas",
"Customer_zip": 13498, "Customer_birth_year": 1981,
"Customer_gender": "F", "Customer_newsletter_sub": true,
"Customer_membership_status": false,
"Customer_credit_status": false}'
"OK"
redis> SET payment:2 '{"Customer_id": 172,
"Transaction_number": "396589804-7", "Payment_date":
"28-9-2020", "Payment_is_late": false}'
"OK"
```

```
redis> SET payment:15 '{"Customer_id": 172,
"Transaction_number": "401129380-4", "Payment_date":
"23-8-2015", "Payment_is_late": true}'
"OK"
redis>
```

Q3. Write the first two records into a JSON document and store in MongoDB.

We have created a new database in MongoDB compass with the name of Customer_DBMS and given under it two collections named as customer and payments. Below are attached the screen shots for this step.



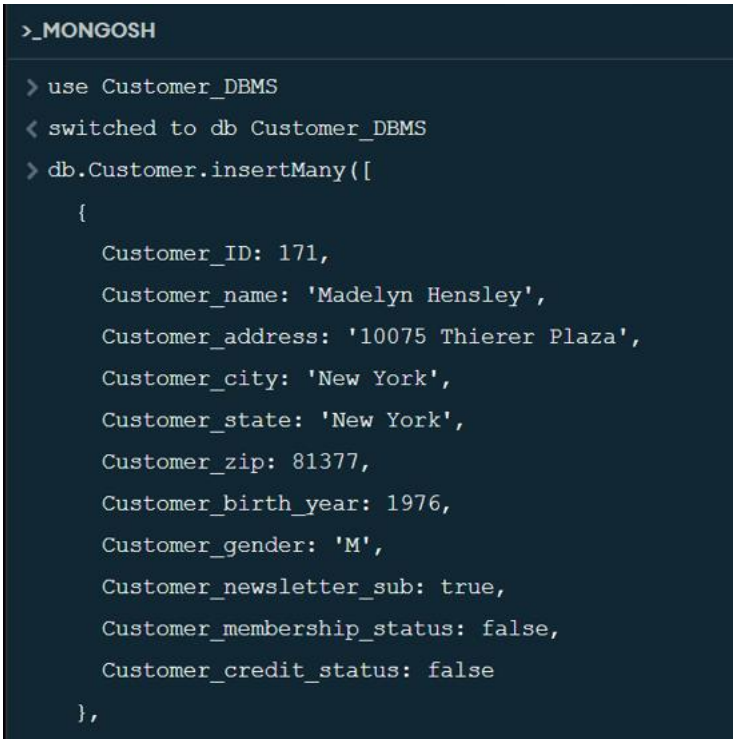


Customer

Now we will record the first 2 records of customer in JSON format in mongosh MongoDB through the code given below. Screenshots have also been attached to show this step.

```
use Customer_DBMS
db.Customer.insertMany([
 {
  Customer_ID: 171,
  Customer_name: 'Madelyn Hensley',
  Customer_address: '10075 Thierer Plaza',
  Customer_city: 'New York',
  Customer_state: 'New York',
  Customer_zip: 81377,
  Customer_birth_year: 1976,
  Customer_gender: 'M',
  Customer_newsletter_sub: true,
  Customer_membership_status: false,
  Customer_credit_status: false
 },
 {
  Customer_ID: 172,
  Customer_name: 'Lonny Foster',
  Customer_address: '23901 Park Meadow Dr',
  Customer_city: 'Austin',
  Customer_state: 'Texas',
  Customer_zip: 13498,
  Customer_birth_year: 1981,
  Customer_gender: 'F',
  Customer_newsletter_sub: true,
```

```
    Customer_membership_status: false,
    Customer_credit_status: false
  }
]);
```



Payment

Now we will record the first 2 records of payment in JSON format in mongosh MongoDB through the code given below. Screenshots have also been attached to show this step.

```
use Customer_DBMS
db.Payment.insertMany([
 {
  Payment_id: 1,
  Customer_id: 181,
  Transaction_number: '146268743-8',
  Payment_date: '23-8-2020',
  Payment_is_late: true
 },
 {
  Payment_id: 2,
  Customer_id: 172,
  Transaction_number: '396589804-7',
  Payment_date: '28-9-2020',
  Payment_is_late: false
 }
]);
```

>_MONGOSH

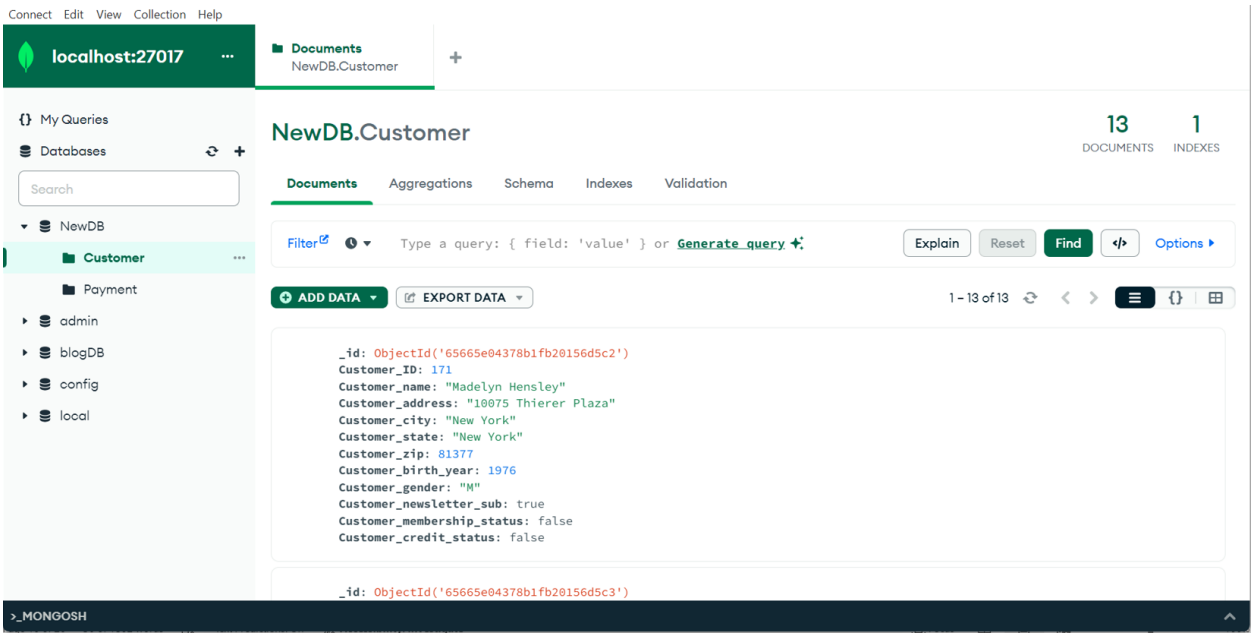
```
> db.Payment.insertMany([
    {
      Payment_id: 1,
      Customer_id: 181,
      Transaction_number: '146268743-8',
      Payment_date: '23-8-2020',
      Payment_is_late: true
    },
    {
      Payment_id: 2,
      Customer_id: 172,
      Transaction_number: '396589804-7',
      Payment_date: '28-9-2020',
      Payment_is_late: false
    }
  ]);
```

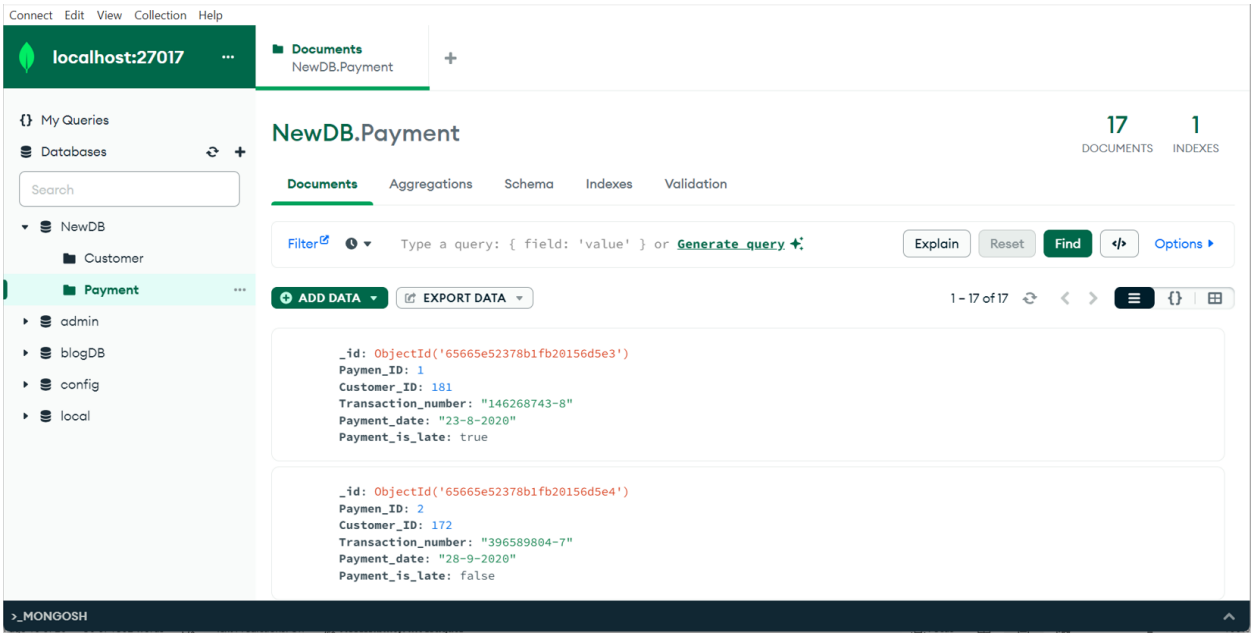
>_MONGOSH

```
    },
    {
      Payment_id: 2,
      Customer_id: 172,
      Transaction_number: '396589804-7',
      Payment_date: '28-9-2020',
      Payment_is_late: false
    }
  ]);

< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("6563ff6055f0eb89de87c5bc"),
      '1': ObjectId("6563ff6055f0eb89de87c5bd")
    }
  }
```

Q4. Delete the collection(s) from point 3, rewrite them into an XML document, and store again in MongoDB.

Now we have deleted the old collections and database and now have made a new database with the name of NewDB and have added two collections, one with the name of customer and the other with the name of payment. Below is attached screenshot for this step.





Customer

Now we will record the first 2 records of customer in XML format in mongosh MongoDB through the code given below. We must keep in mind that XML is not supported by MongoDB so we must come up with a way where we can save the records in MongoDB but use the code in XML format. For that we have assigned variable names to every data and attributes and this way we can successfully record the data in MongoDB through XML format. Screenshots have also been attached to show this step.

```
use NewDB

db.Customer.insertMany ([

{

   "ID": "<Customer_ID>171</Customer_ID>",

   "Name": "<Customer_name>Madelyn Hensley</Customer_name>",

   "Address": "<Customer_address>10075 Thierer Plaza</Customer_address>",

   "City": "<Customer_city>New York</Customer_city>",

   "State": "<Customer_state>New York</Customer_state>",

   "Zipcode": "<Customer_zip>81377</Customer_zip>",

   "Birthyear": "<Customer_birth_year>1976</Customer_birth_year>",

   "Gender": "<Customer_gender>M</Customer_gender>",

   "Subscription": "<Customer_newsletter_sub>TRUE</Customer_newsletter_sub>",

   "Membership": "<Customer_membership_status>FALSE</Customer_membership_status>",

   "Creditstatus": "<Customer_credit_status>FALSE</Customer_credit_status>"

},

{

   "ID": "<Customer_ID>172</Customer_ID>",

   "Name": "<Customer_name>Lonny Foster</Customer_name>",

   "Address": "<Customer_address>23901 Park Meadow Dr</Customer_address>",

   "City": "<Customer_city>Austin</Customer_city>",

   "State": "<Customer_state>Texas</Customer_state>",

   "Zipcode": "<Customer_zip>13498</Customer_zip>",

   "Birthyear": "<Customer_birth_year>1981</Customer_birth_year>",
```

"Gender": "<Customer_gender>F</Customer_gender>",

"Subscription": "<Customer_newsletter_sub>TRUE</Customer_newsletter_sub>",

"Membership": "<Customer_membership_status>FALSE</Customer_membership_status>",

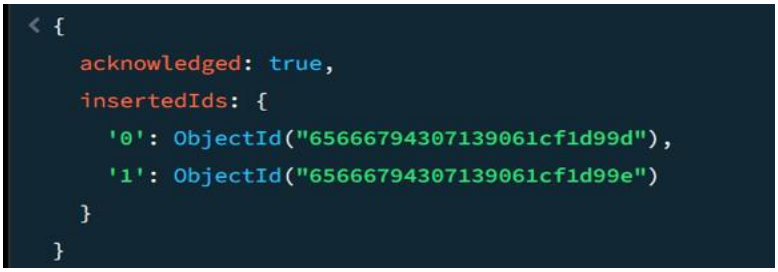
"Creditstatus": "<Customer_credit_status>FALSE</Customer_credit_status>"

}

])

```
> use NewDB
< switched to db NewDB
> db.Customer.insertMany ([
  {
      "ID": "<Customer_ID>171</Customer_ID>",
      "Name": "<Customer_name>Madelyn Hensley</Customer_name>",
      "Address": "<Customer_address>10075 Thierer Plaza</Customer_address>",
      "City": "<Customer_city>New York</Customer_city>",
      "State": "<Customer_state>New York</Customer_state>",
      "Zipcode": "<Customer_zip>81377</Customer_zip>",
      "Birthyear": "<Customer_birth_year>1976</Customer_birth_year>",
      "Gender": "<Customer_gender>M</Customer_gender>",
      "Subscription": "<Customer_newsletter_sub>TRUE</Customer_newsletter_sub>",
      "Membership": "<Customer_membership_status>FALSE</Customer_membership_status>",
      "Creditstatus": "<Customer_credit_status>FALSE</Customer_credit_status>"
  },
```

```
>_MONGOSH
  {
      "ID": "<Customer_ID>172</Customer_ID>",
      "Name": "<Customer_name>Lonny Foster</Customer_name>",
      "Address": "<Customer_address>23901 Park Meadow Dr</Customer_address>",
      "City": "<Customer_city>Austin</Customer_city>",
      "State": "<Customer_state>Texas</Customer_state>",
      "Zipcode": "<Customer_zip>13498</Customer_zip>",
      "Birthyear": "<Customer_birth_year>1981</Customer_birth_year>",
      "Gender": "<Customer_gender>F</Customer_gender>",
      "Subscription": "<Customer_newsletter_sub>TRUE</Customer_newsletter_sub>",
      "Membership": "<Customer_membership_status>FALSE</Customer_membership_status>",
      "Creditstatus": "<Customer_credit_status>FALSE</Customer_credit_status>"
  }
  ])
```



Payment

Now we will record the first 2 records of payment data in XML format in mongosh MongoDB through the code given below. Same way as before we must keep in mind that XML is not supported by MongoDB so we must come up with a way where we can save the records in MongoDB but use the code in XML format. For that we have assigned variable names to every data and attributes and this way we can successfully record the data in MongoDB through XML format. Screenshots have also been attached to show this step.

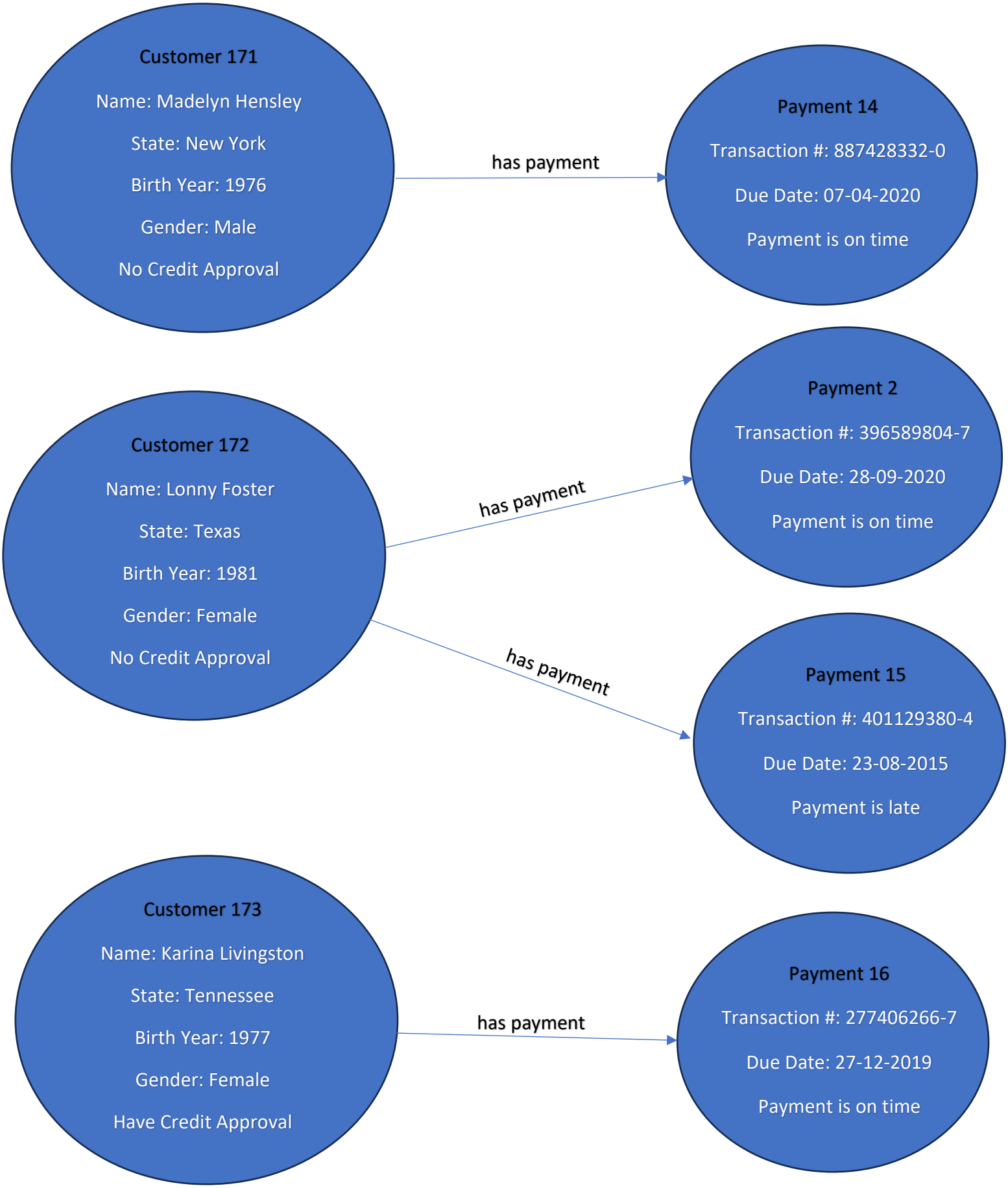
```
use NewDB

db.Payment.insertMany ([

{

   "PaymentID": "<Payment_id>1</Payment_id>",

   "CustomerID": "<Customer_id>181</Customer_id>",

   "Transaction#": "<Transaction_number>146268743-8</Transaction_number>",

   "PaymentDate": "<Payment_date>23-8-2020</Payment_date>",

   "Latepayment": "<Payment_is_late>TRUE</Payment_is_late>"

},

{

   "PaymentID": "<Payment_id>2</Payment_id>",

   "CustomerID": "<Customer_id>172</Customer_id>",

   "Transaction#": "<Transaction_number>396589804-7</Transaction_number>",

   "PaymentDate": "<Payment_date>28-9-2020</Payment_date>",

   "Latepayment": "<Payment_is_late>FALSE</Payment_is_late>"

}

])
```

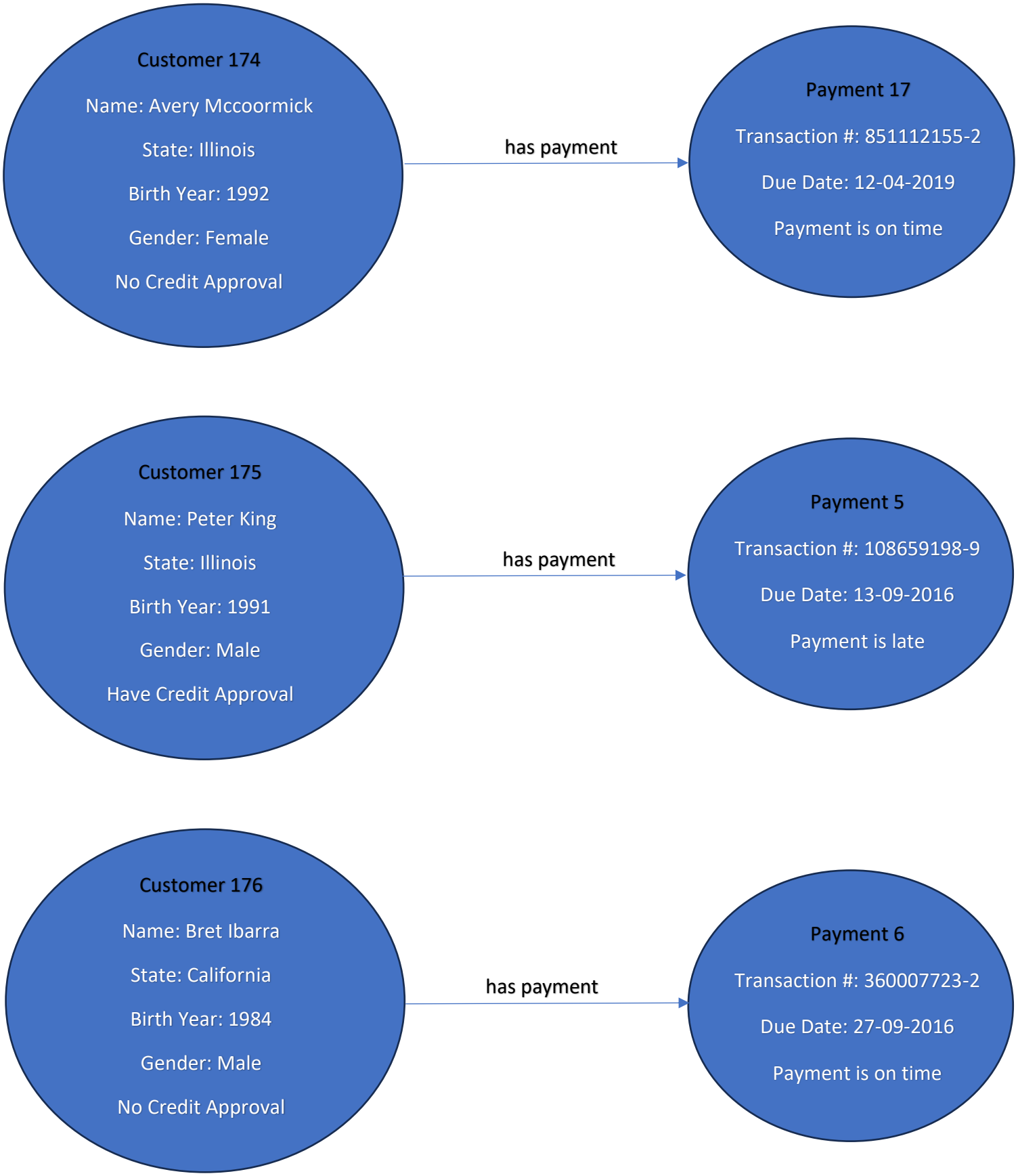
```
>_MONGOSH

> db.Payment.insertMany ([
  {
      "PaymentID": "<Payment_id>1</Payment_id>",
      "CustomerID": "<Customer_id>181</Customer_id>",
      "Transaction#": "<Transaction_number>146268743-8</Transaction_number>",
      "PaymentDate": "<Payment_date>23-8-2020</Payment_date>",
      "Latepayment": "<Payment_is_late>TRUE</Payment_is_late>"
  },
  {
      "PaymentID": "<Payment_id>2</Payment_id>",
      "CustomerID": "<Customer_id>172</Customer_id>",
      "Transaction#": "<Transaction_number>396589804-7</Transaction_number>",
      "PaymentDate": "<Payment_date>28-9-2020</Payment_date>",
      "Latepayment": "<Payment_is_late>FALSE</Payment_is_late>"
  }
  ])
```

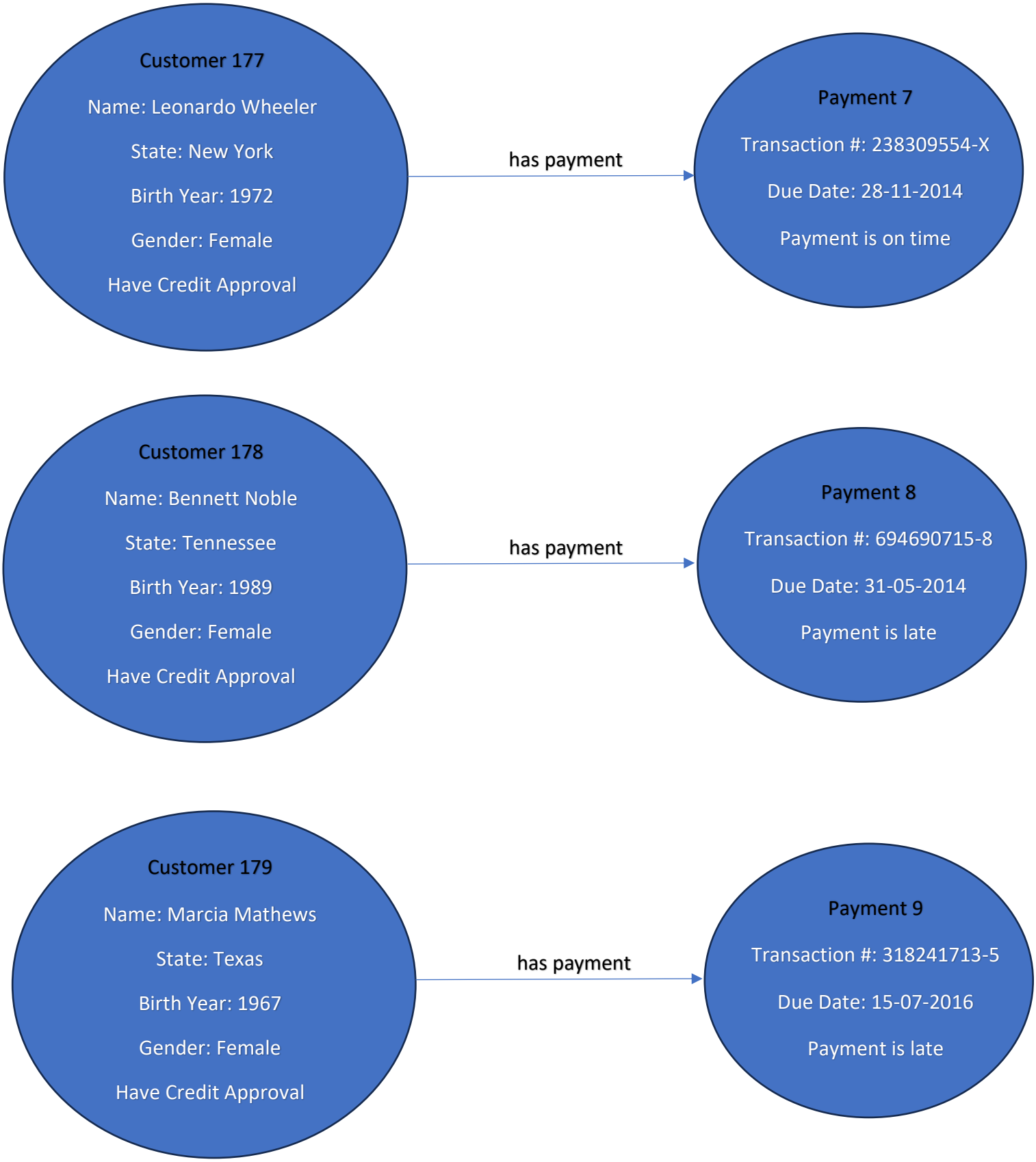
```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId("656669ad307139061cf1d99f"),
      '1': ObjectId("656669ad307139061cf1d9a0")
    }
  }
```

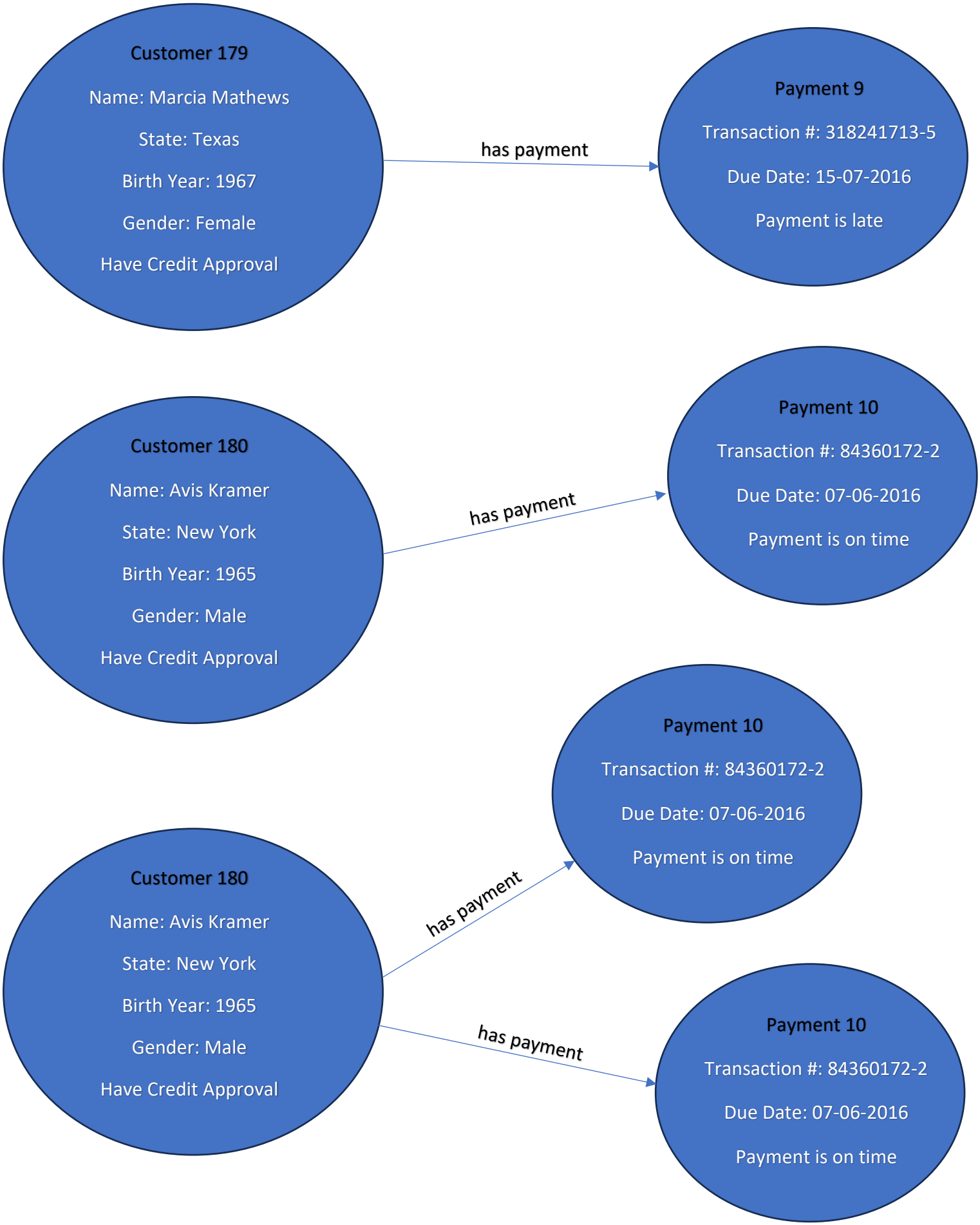
Q5. Design a graph-based diagram. Use nodes to represent entities and edges to represent relationships.

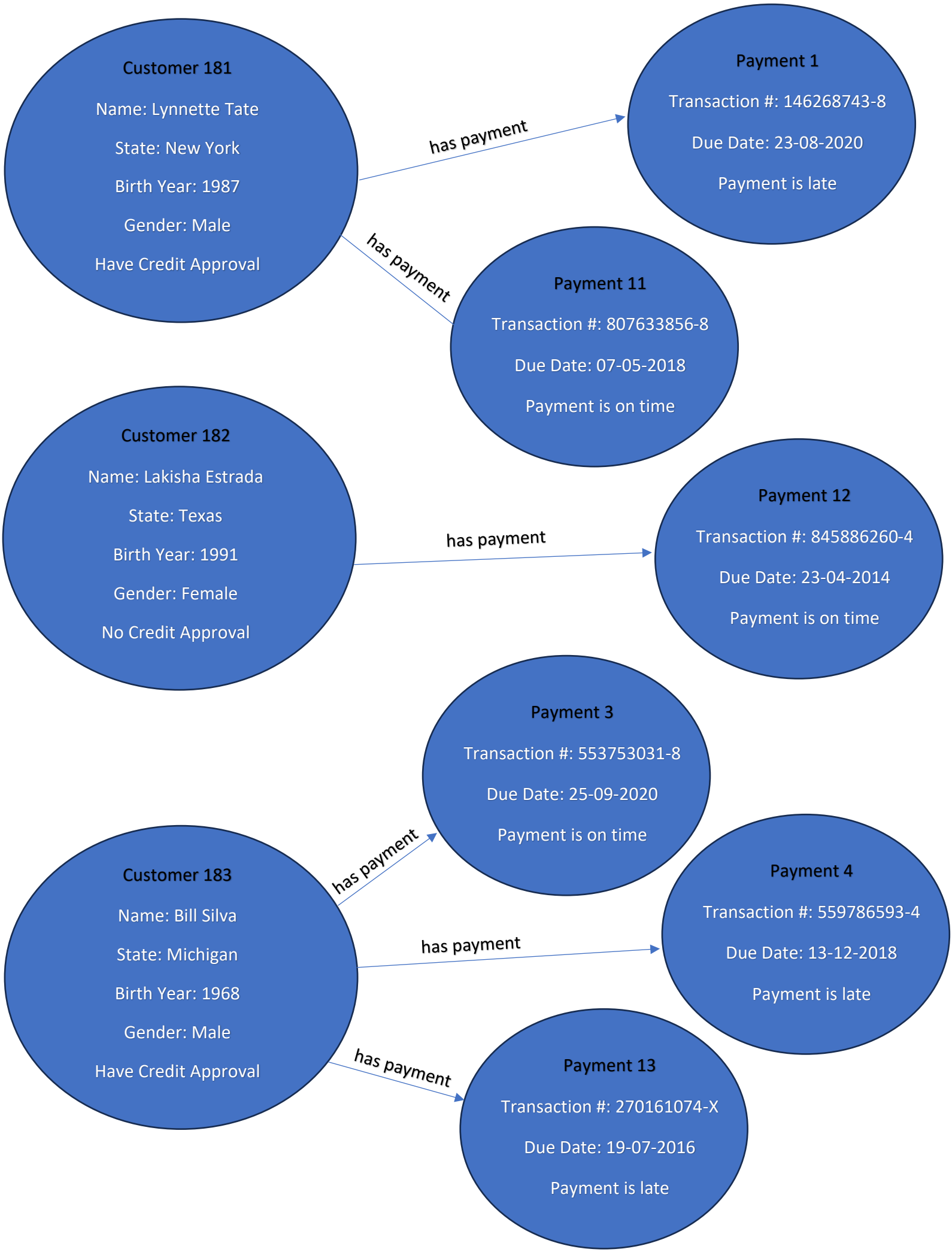
We have made a graph-based diagram as below where we have two nodes, one for customer and the other for payment. We have added 6 attributes for customers in the customer node that shows customer id, name, state, birth year, gender, and credit approval status. Then in the payment node we have added four nodes mainly as payment id, transaction number, due date and finally the payment status if it is delayed for a specific customer or not. We have connected payments with each respective customer through customer id being a common factor between both entities and this relationship is shown by the edges in the graph-based diagram.


Customer 171
Name: Madelyn Hensley
State: New York
Birth Year: 1976
Gender: Male
No Credit Approval
has payment
Payment 14
Transaction #: 887428332-0
Due Date: 07-04-2020
Payment is on time
Customer 172
Name: Lonny Foster
State: Texas
Birth Year: 1981
Gender: Female
No Credit Approval
has payment
Payment 2
Transaction #: 396589804-7
Due Date: 28-09-2020
Payment is on time
has payment
Payment 15
Transaction #: 401129380-4
Due Date: 23-08-2015
Payment is late
Customer 173
Name: Karina Livingston
State: Tennessee
Birth Year: 1977
Gender: Female
Have Credit Approval
has payment
Payment 16
Transaction #: 277406266-7
Due Date: 27-12-2019
Payment is on time


Customer 174
Name: Avery Mccoormick
State: Illinois
Birth Year: 1992
Gender: Female
No Credit Approval
has payment
Payment 17
Transaction #: 851112155-2
Due Date: 12-04-2019
Payment is on time
Customer 175
Name: Peter King
State: Illinois
Birth Year: 1991
Gender: Male
Have Credit Approval
has payment
Payment 5
Transaction #: 108659198-9
Due Date: 13-09-2016
Payment is late
Customer 176
Name: Bret Ibarra
State: California
Birth Year: 1984
Gender: Male
No Credit Approval
has payment
Payment 6
Transaction #: 360007723-2
Due Date: 27-09-2016
Payment is on time


Customer 177
Name: Leonardo Wheeler
State: New York
Birth Year: 1972
Gender: Female
Have Credit Approval
has payment
Payment 7
Transaction #: 238309554-X
Due Date: 28-11-2014
Payment is on time
Customer 178
Name: Bennett Noble
State: Tennessee
Birth Year: 1989
Gender: Female
Have Credit Approval
has payment
Payment 8
Transaction #: 694690715-8
Due Date: 31-05-2014
Payment is late
Customer 179
Name: Marcia Mathews
State: Texas
Birth Year: 1967
Gender: Female
Have Credit Approval
has payment
Payment 9
Transaction #: 318241713-5
Due Date: 15-07-2016
Payment is late


Customer 179
Name: Marcia Mathews
State: Texas
Birth Year: 1967
Gender: Female
Have Credit Approval
has payment
Payment 9
Transaction #: 318241713-5
Due Date: 15-07-2016
Payment is late
Customer 180
Name: Avis Kramer
State: New York
Birth Year: 1965
Gender: Male
Have Credit Approval
has payment
Payment 10
Transaction #: 84360172-2
Due Date: 07-06-2016
Payment is on time
Customer 180
Name: Avis Kramer
State: New York
Birth Year: 1965
Gender: Male
Have Credit Approval
has payment
Payment 10
Transaction #: 84360172-2
Due Date: 07-06-2016
Payment is on time
has payment
Payment 10
Transaction #: 84360172-2
Due Date: 07-06-2016
Payment is on time


Customer 181
Name: Lynnette Tate
State: New York
Birth Year: 1987
Gender: Male
Have Credit Approval
has payment
Payment 1
Transaction #: 146268743-8
Due Date: 23-08-2020
Payment is late
has payment
Payment 11
Transaction #: 807633856-8
Due Date: 07-05-2018
Payment is on time
Customer 182
Name: Lakisha Estrada
State: Texas
Birth Year: 1991
Gender: Female
No Credit Approval
has payment
Payment 12
Transaction #: 845886260-4
Due Date: 23-04-2014
Payment is on time
Payment 3
Transaction #: 553753031-8
Due Date: 25-09-2020
Payment is on time
Customer 183
Name: Bill Silva
State: Michigan
Birth Year: 1968
Gender: Male
Have Credit Approval
has payment
Payment 4
Transaction #: 559786593-4
Due Date: 13-12-2018
Payment is late
has payment
has payment
Payment 13
Transaction #: 270161074-X
Due Date: 19-07-2016
Payment is late

Q6. Use create statements to implement the nodes and relationships for the diagram you have designed in Neo4j.

In this part we have basically created nodes for all the customers that include the detail for their attributes as well but node is mainly displayed by the customer name. Then we created payment node and included all its attributes and payment node is displayed by payment id. Then we used match function to connect each payment with the respective customer and this is similar to what we have done in the previous part and we have replicated that in Neo4j. Attached below is the code to execute this and also screenshot have been posted below to show how the graph diagram looks in Neo4j and how the edges represent the function between two nodes like in this case it shows that customer shown by customer name made bill which is shown by the payment id.

Customer

CREATE (:Customer {id: 171, name: 'Madelyn Hensley', city: 'New York', state: 'New York', birth_year: 1976, gender: 'M', credit_status: FALSE})

CREATE (:Customer {id: 172, name: 'Lonny Foster', city: 'Austin', state: 'Texas', birth_year: 1981, gender: 'F', credit_status: FALSE})

CREATE (:Customer {id: 173, name: 'Karina Livingston', city: 'Chattanooga', state: 'Tennessee', birth_year: 1977, gender: 'F', credit_status: TRUE})

CREATE (:Customer {id: 174, name: 'Avery Mccormick', city: 'Chicago', state: 'Illinois', birth_year: 1992, gender: 'F', credit_status: FALSE})

CREATE (:Customer {id: 175, name: 'Peter King', city: 'Chicago', state: 'Illinois', birth_year: 1991, gender: 'M', credit_status: TRUE})

CREATE (:Customer {id: 176, name: 'Bret Ibarra', city: 'San Diego', state: 'California', birth_year: 1984, gender: 'M', credit_status: FALSE})

CREATE (:Customer {id: 177, name: 'Leonardo Wheeler', city: 'New York', state: 'New York', birth_year: 1972, gender: 'F', credit_status: TRUE})

CREATE (:Customer {id: 178, name: 'Bennett Noble', city: 'Hixson', state: 'Tennessee', birth_year: 1989, gender: 'F', credit_status: TRUE})

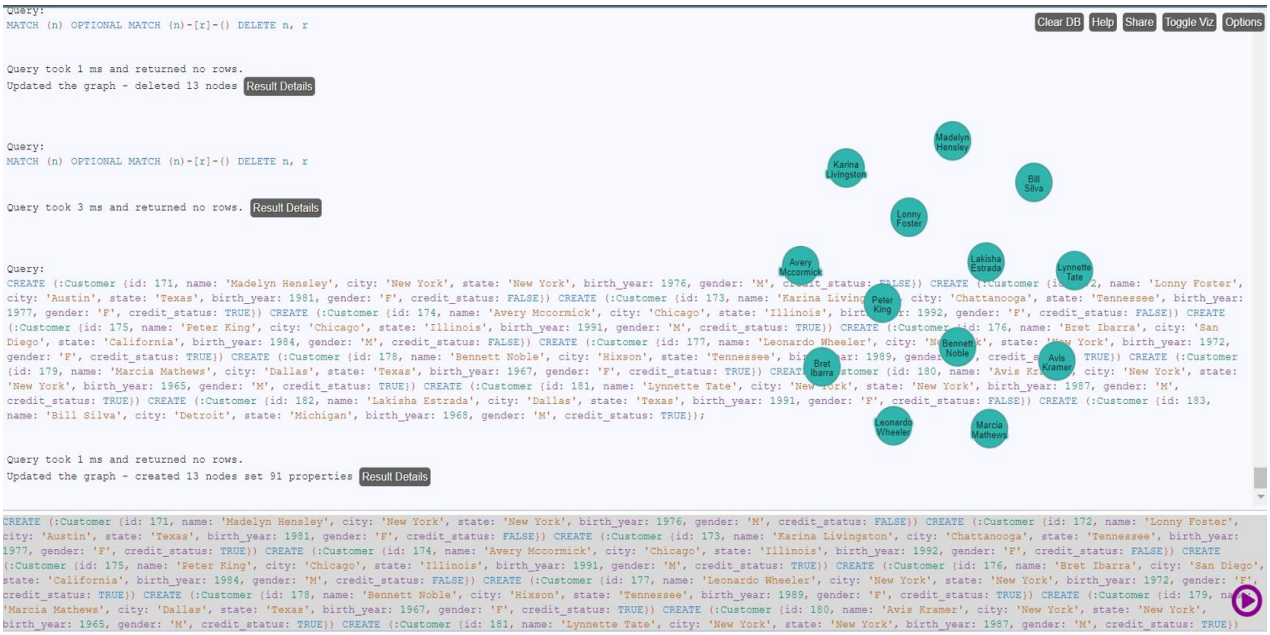
CREATE (:Customer {id: 179, name: 'Marcia Mathews', city: 'Dallas', state: 'Texas', birth_year: 1967, gender: 'F', credit_status: TRUE})

CREATE (:Customer {id: 180, name: 'Avis Kramer', city: 'New York', state: 'New York', birth_year: 1965, gender: 'M', credit_status: TRUE})

CREATE (:Customer {id: 181, name: 'Lynnette Tate', city: 'New York', state: 'New York', birth_year: 1987, gender: 'M', credit_status: TRUE})

CREATE (:Customer {id: 182, name: 'Lakisha Estrada', city: 'Dallas', state: 'Texas', birth_year: 1991, gender: 'F', credit_status: FALSE})

CREATE (:Customer {id: 183, name: 'Bill Silva', city: 'Detroit', state: 'Michigan', birth_year: 1968, gender: 'M', credit_status: TRUE});



Payment and Match function

MATCH (c:Customer {id: 171})

CREATE (p:Payment {Payment_ID: 1, transaction_id: '146268743-8', date: '23-8-2020', status: TRUE})-[:MADE]->(c)

MATCH (c:Customer {id: 172})

CREATE (p:Payment {Payment_ID: 2, transaction_id: '396589804-7', date: '28-9-2020', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 183})

CREATE (p:Payment {Payment_ID: 3, transaction_id: '553753031-8', date: '25-9-2020', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 183})

CREATE (p:Payment {Payment_ID: 4, transaction_id: '559786593-4', date: '13-12-2018', status: TRUE})-[:MADE]->(c)

MATCH (c:Customer {id: 175})

CREATE (p:Payment {Payment_ID: 5, transaction_id: '108659198-9', date: '13-9-2016', status: TRUE})-[:MADE]->(c)

MATCH (c:Customer {id: 176})

CREATE (p:Payment {Payment_ID: 6, transaction_id: '360007723-2', date: '27-9-2016', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 177})

CREATE (p:Payment {Payment_ID: 7, transaction_id: '238309554-X', date: '28-11-2014', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 178})

CREATE (p:Payment {Payment_ID: 8, transaction_id: '694690715-8', date: '31-5-2014', status: TRUE})-[:MADE]->(c)

MATCH (c:Customer {id: 179})

CREATE (p:Payment {Payment_ID: 9, transaction_id: '318241713-5', date: '15-7-2016', status: TRUE})-[:MADE]->(c)

MATCH (c:Customer {id: 180})

CREATE (p:Payment {Payment_ID: 10, transaction_id: '84360172-2', date: '6-7-2016', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 181})

CREATE (p:Payment {Payment_ID: 11, transaction_id: '807633856-8', date: '5-7-2018', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 182})

CREATE (p:Payment {Payment_ID: 12, transaction_id: '845886260-4', date: '23-4-2014', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 183})

CREATE (p:Payment {Payment_ID: 13, transaction_id: '270161074-X', date: '19-7-2016', status: TRUE})-[:MADE]->(c)

MATCH (c:Customer {id: 171})

CREATE (p:Payment {Payment_ID: 14, transaction_id: '887428332-0', date: '4-7-2020', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 172})

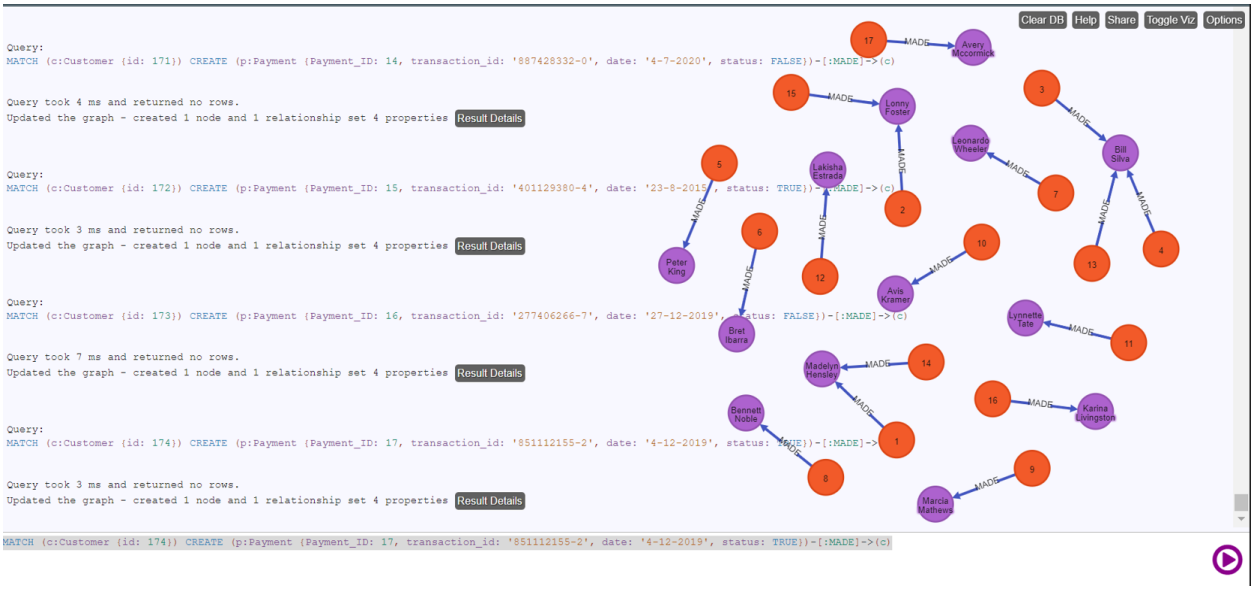
CREATE (p:Payment {Payment_ID: 15, transaction_id: '401129380-4', date: '23-8-2015', status: TRUE})-[:MADE]->(c)

MATCH (c:Customer {id: 173})

CREATE (p:Payment {Payment_ID: 16, transaction_id: '277406266-7', date: '27-12-2019', status: FALSE})-[:MADE]->(c)

MATCH (c:Customer {id: 174})

CREATE (p:Payment {Payment_ID: 17, transaction_id: '851112155-2', date: '4-12-2019', status: TRUE})-[:MADE]->(c)


Clear DB
Help
Share
Toggle Viz
Options
Query:
MATCH (c:Customer {id: 171}) CREATE (p:Payment {Payment_ID: 14, transaction_id: '887428332-0', date: '4-7-2020', status: FALSE})-[:MADE]->(c)
Query took 4 ms and returned no rows.
Updated the graph - created 1 node and 1 relationship set 4 properties
Result Details
Query:
MATCH (c:Customer {id: 172}) CREATE (p:Payment {Payment_ID: 15, transaction_id: '401129380-4', date: '23-8-2015', status: TRUE})-[:MADE]->(c)
Query took 3 ms and returned no rows.
Updated the graph - created 1 node and 1 relationship set 4 properties
Result Details
Query:
MATCH (c:Customer {id: 173}) CREATE (p:Payment {Payment_ID: 16, transaction_id: '277406266-7', date: '27-12-2019', status: FALSE})-[:MADE]->(c)
Query took 7 ms and returned no rows.
Updated the graph - created 1 node and 1 relationship set 4 properties
Result Details
Query:
MATCH (c:Customer {id: 174}) CREATE (p:Payment {Payment_ID: 17, transaction_id: '851112155-2', date: '4-12-2019', status: TRUE})-[:MADE]->(c)
Query took 3 ms and returned no rows.
Updated the graph - created 1 node and 1 relationship set 4 properties
Result Details
MADE
Avery Mccormick
Lonny Foster
Lakisha Estrada
Leonardo Wheeler
Bill Silva
Peter King
Avis Kramer
Bret Ibarra
Lynnette Tate
Madelyn Hensley
Bennett Noble
Karina Livingston
Marcia Mathews