**Banking Enterprise DBMS**

(Database Modeling in E-R Diagram

&

Database creation in Oracle, MongoDB, Neo4j)

* Case Study done by…

SHARON SAM S (215229137)

VIMAL S E (215229144)

**Abstract of a Banking Environment:**

All these ages money plays an essential role in every one’s life and we live with our daily dealings with the notes and coins. Banking management system helps us to make our daily dealings aesthetic, easy and smooth with money. Banks provide different sort of services to its customers while keeping in with their own daily tasks. And to achieve this task, Banking Enterprises makes a database management system to solve its day today tasks.

The bank management system is an application for maintaining many types of accounts such as savings account for personal uses, current accounts for business people, joint accounts in a bank and each customer can hold multiple accounts but have atleast one account in a branch of a bank.

Further the bank provides different types of loans such as educational loans, business loans, gold loans, home loans, etc. for its customers and the loans are settled via many Installments for a period of time.

**Problem Statement:**

The Intent of the project is to create a Banking Database Management System.

• We consider three different types of banks where each bank has its own different names.

• Primarily bank is organized into branches. And every branch is situated in different areas and can be uniquely identified by their given IFSC Code. Each of these branches monitors all the accounts and loans given to its customers.

• Each account is identified by its account number and it has the balance of that account. Accounts can be held by a customer having a savings account or current account. In case of more than one customer joint accounts are preferred.

• A loan originates at a particular branch and can be held by one or more customers (joint account). A loan is identified by a unique loan id. For each loan, the bank keeps track of the loan amount and the loan payments.

• Although loans are paid in installments with different installment no, it does not uniquely identify a particular payment among those for all the bank's loans, an installment number does identify a particular payment for a specific loan. The date and amount are recorded for each payment.

• Bank customer has a unique customer identifier as Id their details such as name, phone no., gender, dob, age, etc. A customer can have multiple accounts or a checking or a saving account not only in one bank but other banks as well and could also take loans from the banks.

**E-R Diagram Construction:**

[ER diagram](https://www.geeksforgeeks.org/introduction-of-er-model/) is known as Entity-Relationship diagram. It is a model to represent the database structure in different entities with their attributes and relationships between the entities.

Analyzing the working environment (Banking Enterprise) to show the structure and schema of the database and finding the entities and identifying their attributes and their key attributes and considering the relationship between the entities and providing the cardinality constraints we, construct the E-R Model for Banking Enterprise DBMS.

Basically there are Four Steps included in constructing an E-R Model. They are as follows:

**Step 1 − Identify the Entity sets:**

Bank, Branch, Account, Loan, Installment (Weak Entity), Customer

**Step 2 − Identify the Attributes:**

* Bank − Bank Name, Full Bank Name
* Branch − IFSC Code, Branch Name, Branch Address.
* Account − Account No, Account Type, Account Balance.
* Loan − Loan ID, Loan Type, Loan Amount.
* Installment − Installment No, Installment Date, Installment Amount.
* Customer − Customer ID, Customer Name, DOB, Age, Phone No, Gender, Address.

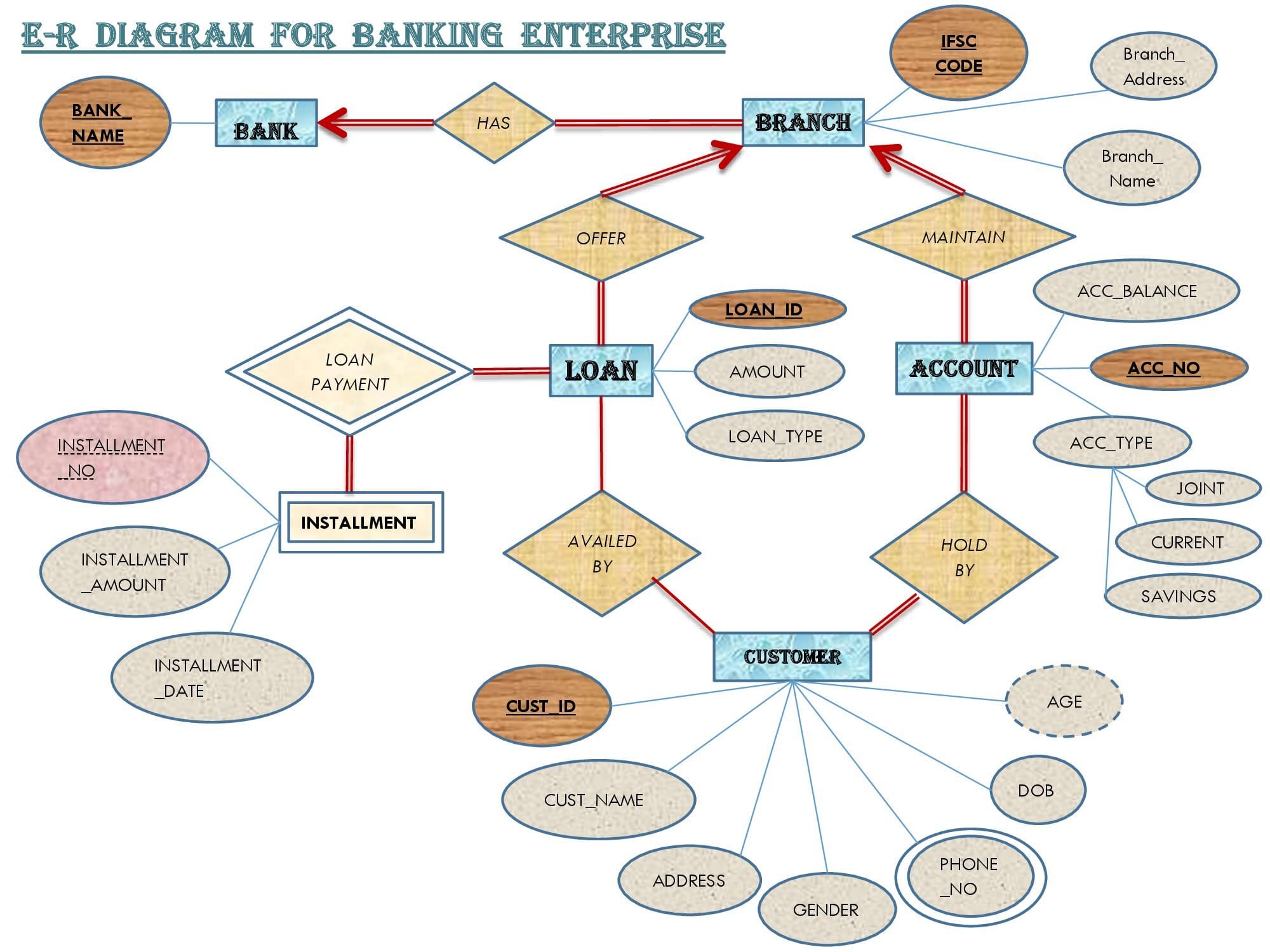
**Step 3 − Identify the Key attributes:**

* Bank Name is the key attribute for Bank entity.
* IFSC Code is the key attribute for Branch entity.
* Account No is the key attribute for Account entity.
* Loan ID is the key attribute for Loan entity.
* Installment No is the key attribute for Installment entity.
* Customer ID is the key attribute for Customer entity.

**Step 4 − Identify the Relationships, Cardinalities & Participations:**

* Bank has Branches => 1 : N (total)
* Branch offer Loans => 1 : N (total)
* Installment payment for Loans => M : N  (total)
* Loan availed by Customer => M : N (partial)
* Branch maintain Accounts => 1 : N (total)
* Account held by Customers => M : N  (total)

**E-R Diagram:**



**RDBMS:**

A relational database is based on the relational model and uses a collection of tables to represent both data and the relationships among those data.

NORMALIZATION:

Another method for designing a relational database is to use a process commonly known as normalization. The goal is to generate a set of relation schemas that allows us to store information without unnecessary redundancy, yet also allows us to retrieve information easily.

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

**SQL for RDBMS in Banking Enterprise:**

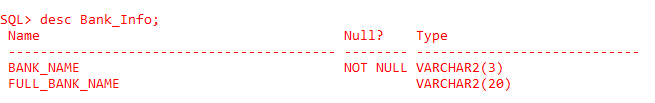
SQL IS Structured Query Language. SQL based database systems for Banking Enterprise could be implemented through Oracle database and the files are imported via SQL Developer.

Database management systems are specifically designed in banks to handle large amounts of data which enables them to store that data, operate on it, and retrieve it when needed, fast enough for their and their customer’s needs. Most banks use a DBMS type called “relational” (RDBMS for short), for good reasons. These reasons are abbreviated ACID (Atomicity, Consistency, Isolation, Durability) properties. Each of the ACID properties helps the banks do their business, which makes relational databases a natural choice for the administration of their customer and account data.

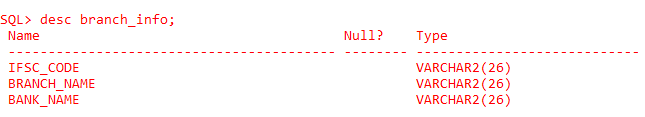
**SQL Queries:**

**Description of the imported table:**

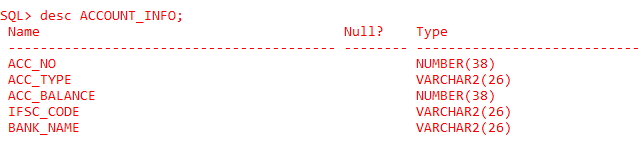
SQL> desc Bank\_Info;



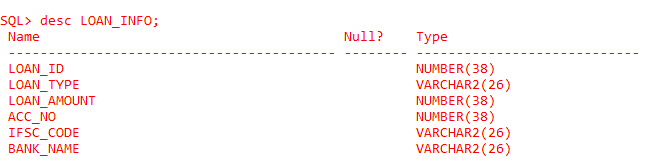
SQL> desc branch\_info;



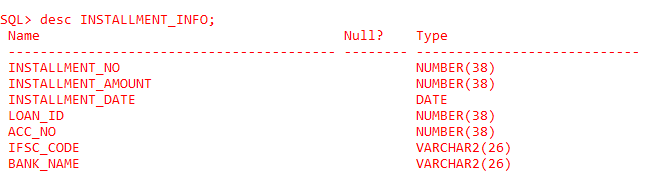
SQL> desc ACCOUNT\_INFO;



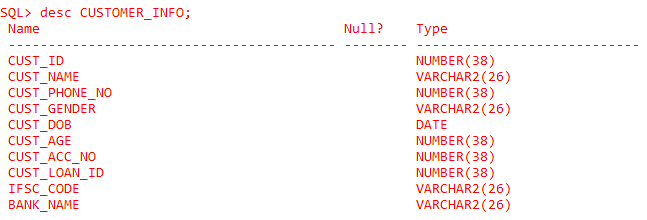
SQL> desc LOAN\_INFO;



SQL> desc INSATLLMENT\_INFO;

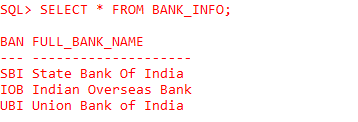


SQL> desc CUSTOMER\_INFO;

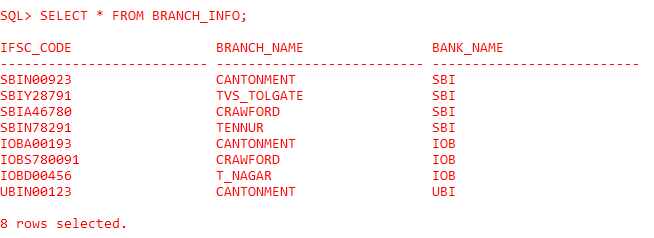


**Read all the entries of the table:**

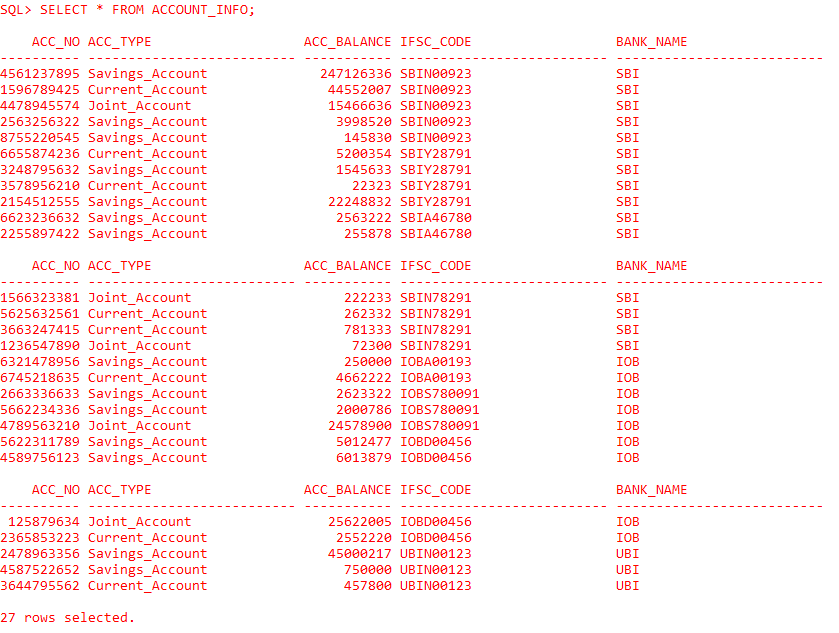
SQL> SELECT \* FROM BANK\_INFO;



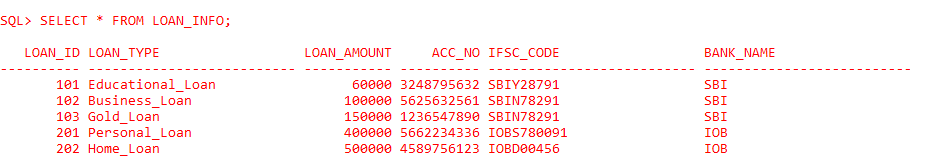
SQL> SELECT \* FROM BRANCH\_INFO;



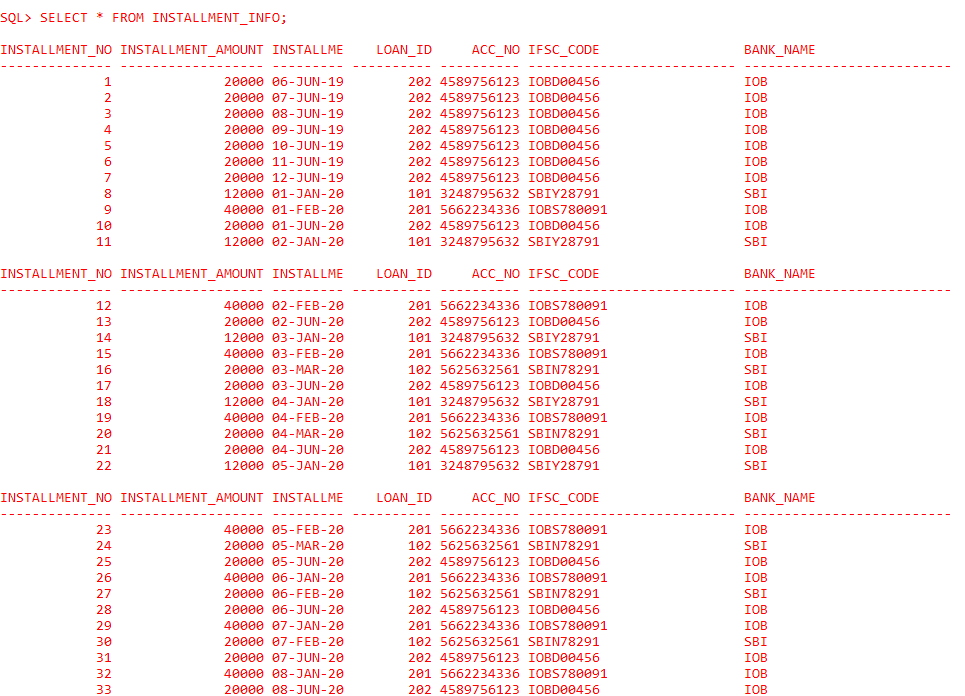
SQL> SELECT \* FROM ACCOUNT\_INFO;

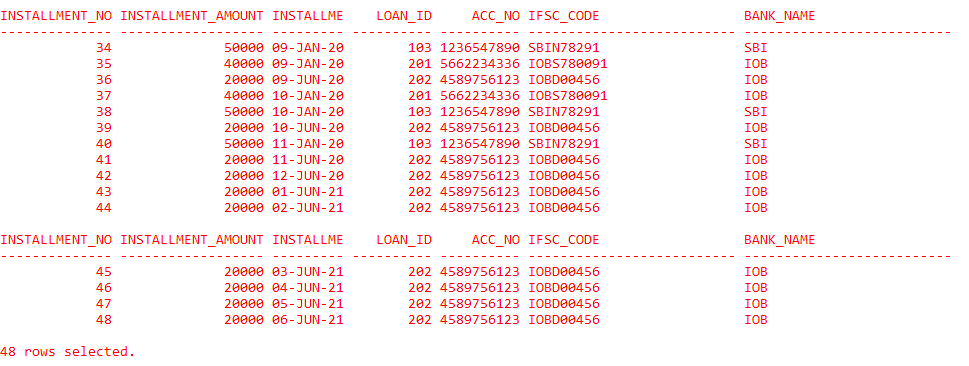


SQL> SELECT \* FROM LOAN\_INFO;

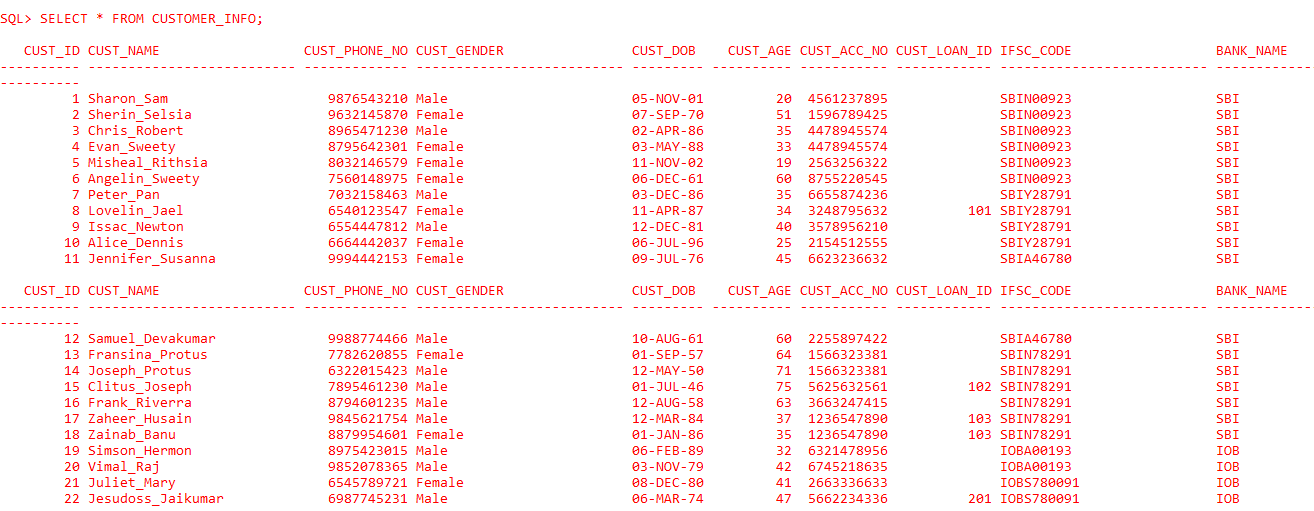


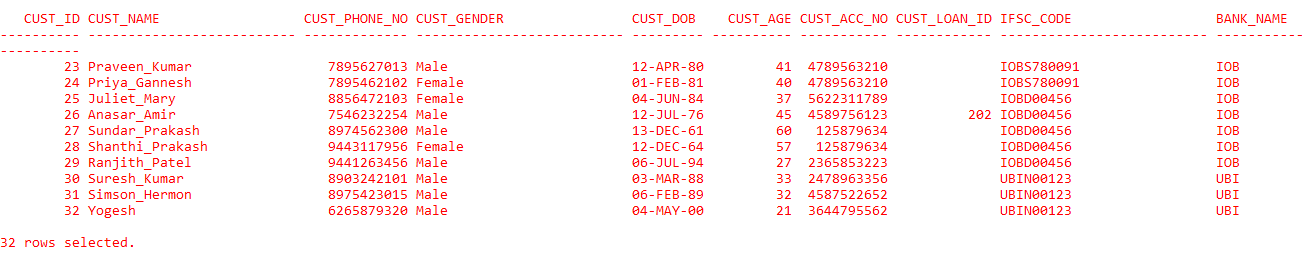
SQL> SELECT \* FROM INSTALLMENT\_INFO;





SQL> SELECT \* FROM CUSTOMER\_INFO;

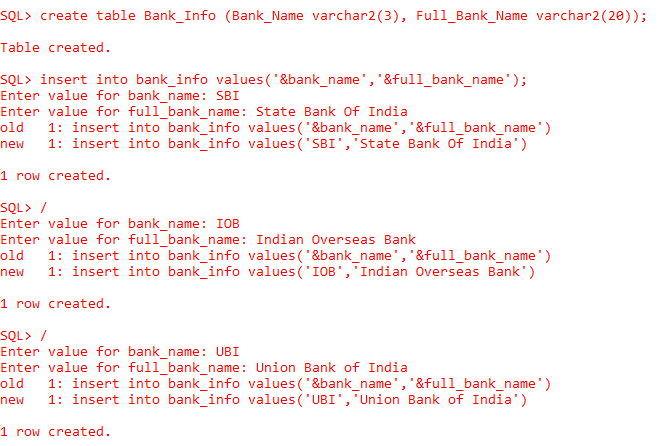




**CURD Operations:**

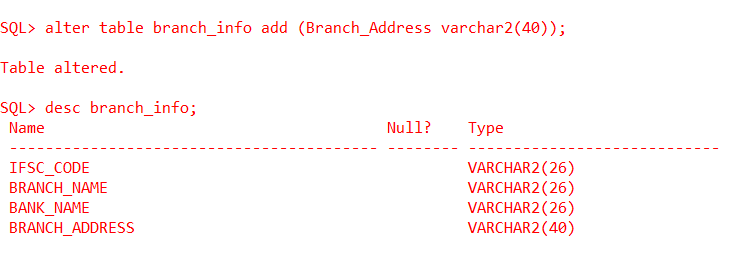
SQL> create table Bank\_Info (Bank\_Name varchar2(3), Full\_Bank\_Name varchar2(20));

SQL> insert into bank\_info values(‘&bank\_name’,’&full\_bank\_name’);

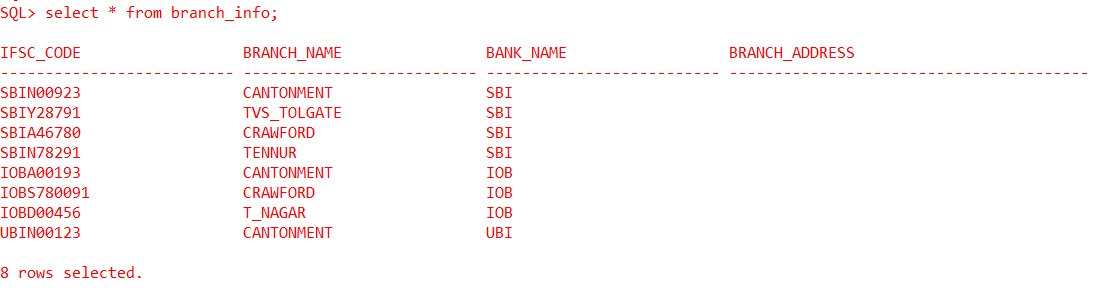


SQL> alter table branch\_info add (Branch\_Address varchar2(40));

SQL> desc branch\_info;

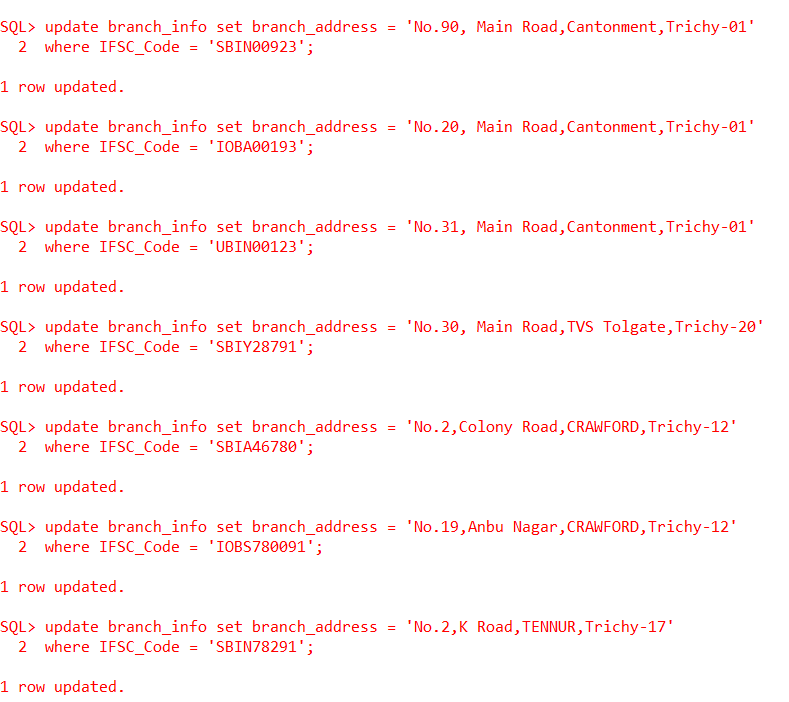


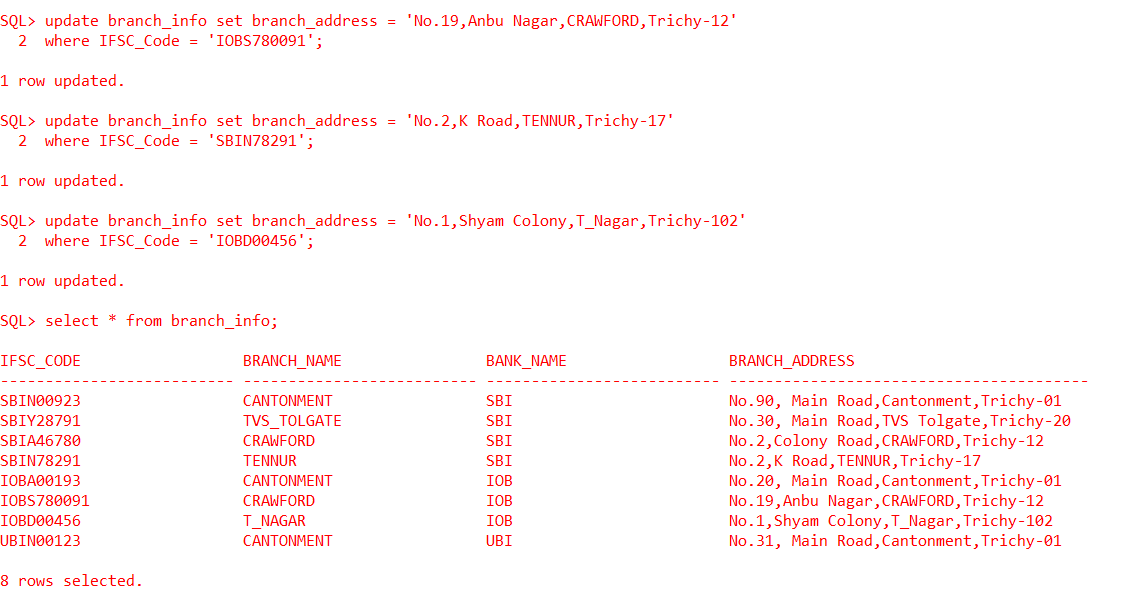
SQL> select \* from branch\_info;



SQL> update branch\_info set branch\_address = ‘No.90, Main Road,Cantonment,Trichy-01’ where IFSC\_Code = ‘SBIN00923’;

………..



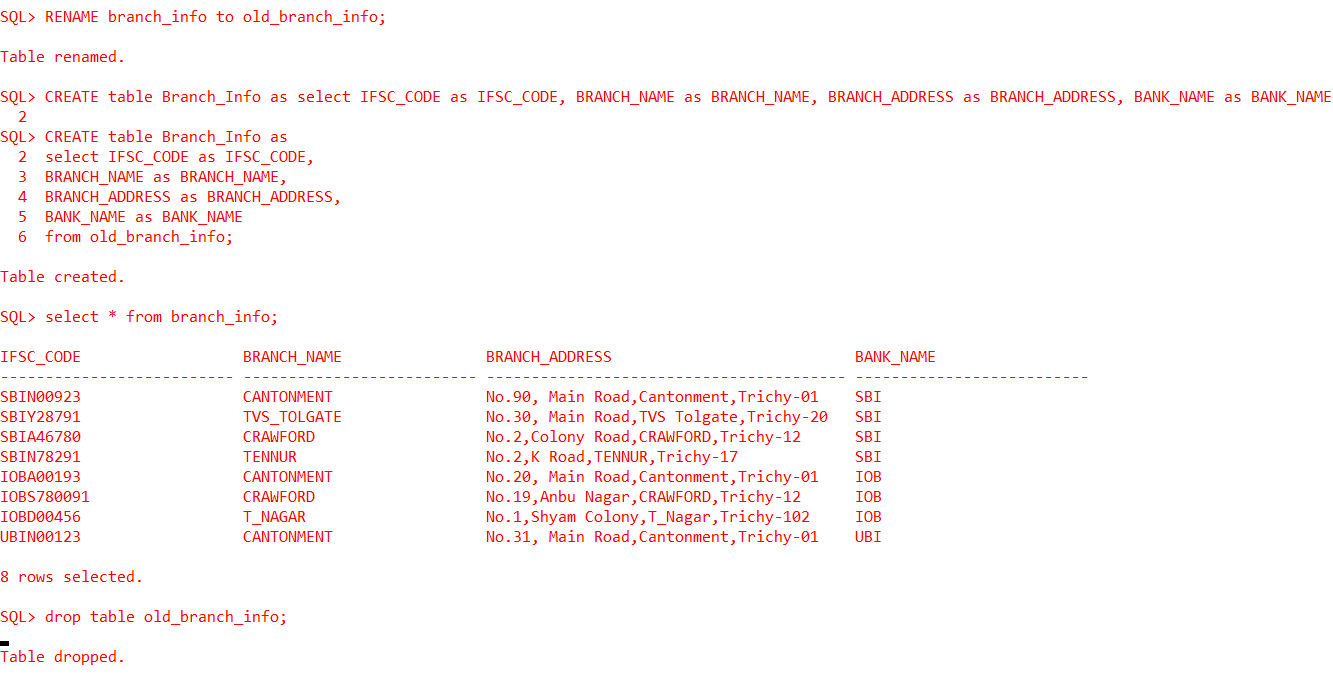


SQL> RENAME branch\_info to old\_branch\_info;

SQL> CREATE table Branch\_Info as select IFSC\_CODE AS IFSC\_CODE, BRANCH\_NAME AS BRANCH\_NAME, BRANCH\_ADDRESS AS BRANCH\_ADDRESS, BANK\_NAME AS BANK\_NAME from old\_branch\_info;

SQL> select \* from branch\_info;

SQL> drop table old\_branch\_info;

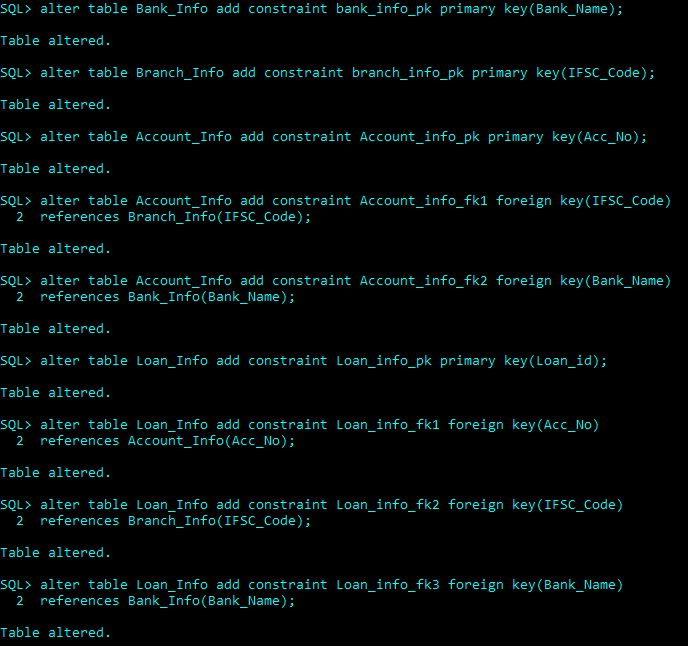


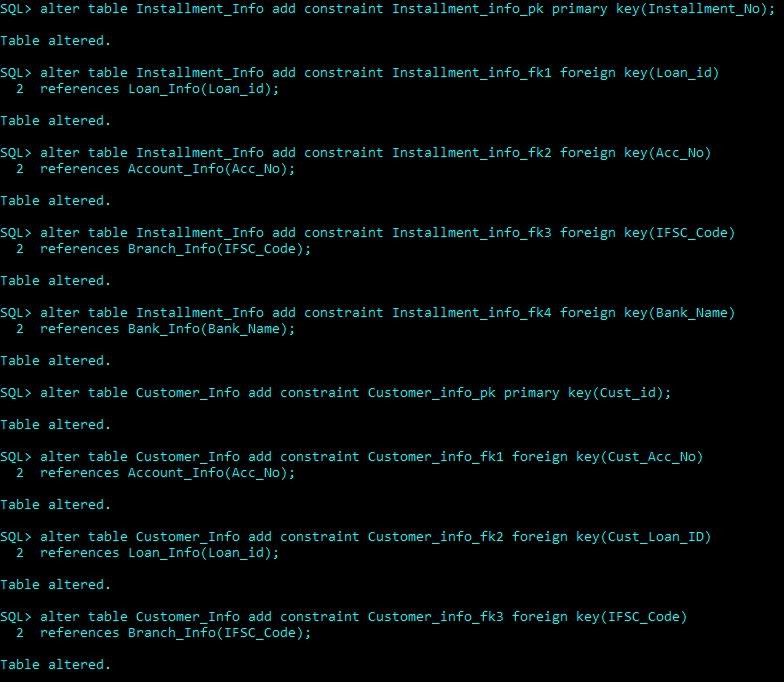
**Primary and Foreign keys constraints:**

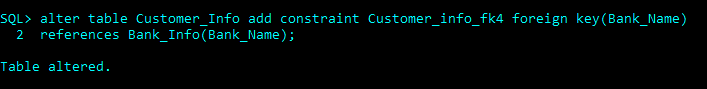
SQL> alter table Bank\_Info add constraint bank\_info\_pk primary key(Bank\_Name);

SQL> alter table Branch\_Info add constraint branch\_info\_pk primary key(IFSC\_Code);

SQL> alter table Branch\_Info add constraint branch\_info\_fk foreign key(Bank\_Name) references Bank\_Info(Bank\_Name);







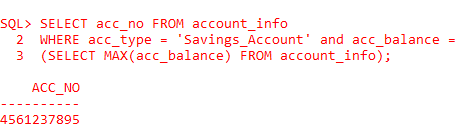
**Subqueries:**

**Showing the Savings Acount which has the highest balance**

SQL> SELECT acc\_no FROM account\_info

2 WHERE acc\_type = 'Savings\_Account' and acc\_balance =

3 (SELECT MAX(acc\_balance) FROM account\_info);

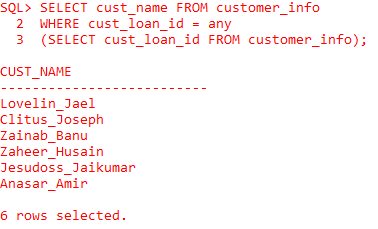


**Showing the customers who have borrowed loans from bank**

SQL> SELECT cust\_name FROM customer\_info

2 WHERE cust\_loan\_id = any

3 (SELECT cust\_loan\_id FROM customer\_info);



**Joins:**

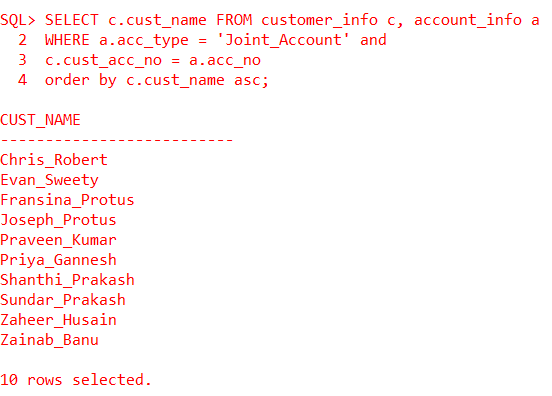
**Showing the customer names who have joint accounts**

SQL> SELECT c.cust\_name FROM customer\_info c, account\_info a

2 WHERE a.acc\_type = 'Joint\_Account' and

3 c.cust\_acc\_no = a.acc\_no

4 order by c.cust\_name asc;



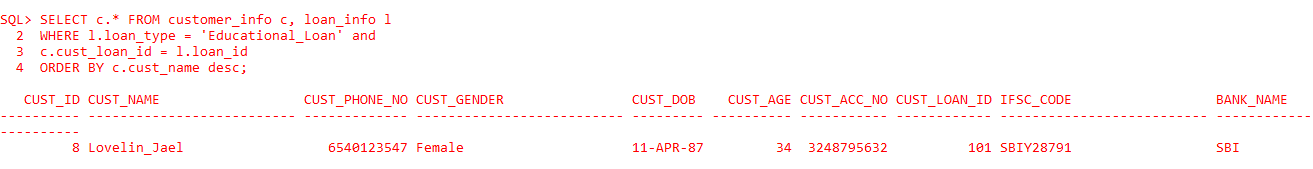
**Showing the details of customer who availed educational loan**

SQL> SELECT c.\* FROM customer\_info c, loan\_info l

2 WHERE l.loan\_type = 'Educational\_Loan' and

3 c.cust\_loan\_id = l.loan\_id

4 ORDER BY c.cust\_name desc;

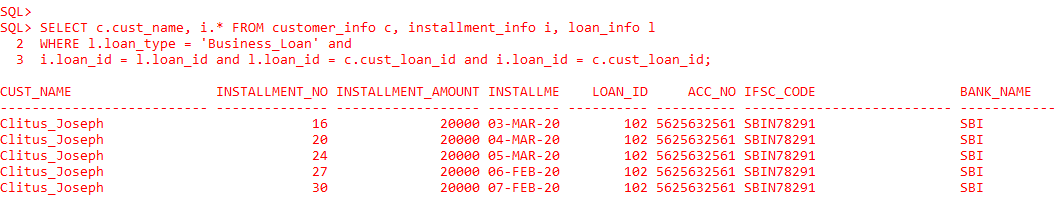


**Showing the installment details of the customer who availed the business loan**

SQL> SELECT c.cust\_name, i.\* FROM customer\_info c, installment\_info i, loan\_info l

2 WHERE l.loan\_type = 'Business\_Loan' and

3 i.loan\_id = l.loan\_id and l.loan\_id = c.cust\_loan\_id and i.loan\_id = c.cust\_loan\_id;



**Subquery & Joins:**

**Showing the customer name and phone no and the account details of the customer who has the lowest minimal balance among the current accounts**

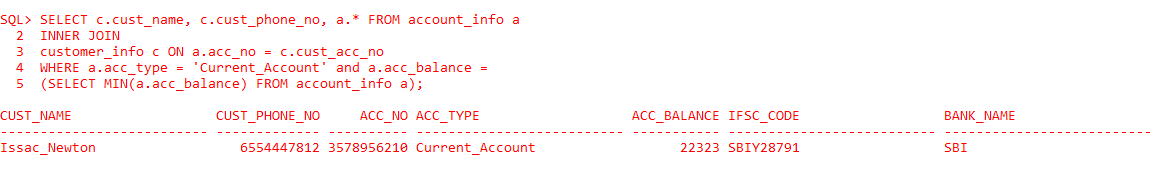
SQL> SELECT c.cust\_name, c.cust\_phone\_no, a.\* FROM account\_info a

2 INNER JOIN

3 customer\_info c ON a.acc\_no = c.cust\_acc\_no

4 WHERE a.acc\_type = 'Current\_Account' and a.acc\_balance =

5 (SELECT MIN(a.acc\_balance) FROM account\_info a);



**NoSQL DBMS:**

NoSQL Database is a non-relational Data Management System that does not require a fixed schema and originally referring to non SQL or non-relational is a database that provides a mechanism for storage and retrieval of data.

As bank enterprises shift to the Digital Economy like using an online transactions which is enabled by cloud, mobile, social media, and big data technologies, the developers and operations teams have to build and maintain web, mobile, and IoT applications faster and faster, and at greater scale. Flexible, high-performance NoSQL is increasingly the preferred database technology to power today’s web, mobile, and IoT applications. With NoSQL, enterprises are better able to both develop with agility and operate at any scale – and deliver the performance and availability required to meet the demands of Digital Economy business.

**MongoDB:**

MongoDB falls in the category of NoSQL document based database. It stores data in documents i.e. BSON documents. BSON is a binary representation of JSON (JavaScript Object Notation) documents. Documents typically store information about one object and data related to that object. Related documents are grouped together in collections. And related collections are stored in a database.

Every document begins and ends with curly braces. Inside the curly braces, an unordered set of field or value pairs that delimited by commas. The values can be of any data types and with all these types available the power of modeling the data as it exists in the real world is aesthetic. Every document has an \_id. The value of \_id must be unique and is immutable.

**NoSQL Queries in MongoDB:**

**CURD Operations:**

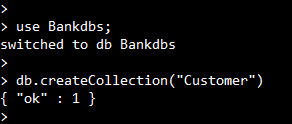
**Creating a Database and a Collection:**

> use Bankdbs;

switched to db Bankdbs

> db.createCollection("Customer")

{ "ok" : 1 }



**Inserting documents into the Collection:**

> db.Customer.insert({\_id:"1","Cust\_Name":"Sharon\_Sam","Cust\_Phone\_No":9876543210,"Cust\_Gender":"Male","Cust\_DOB":"11/05/2001","Cust\_Acc\_Details":{"Acc\_No":4561237895,"Acc\_Type":"Savings\_Account","Acc\_Balance":247126336,"Bank\_Info":{"IFSC\_Code":"SBIN00923","Branch\_Name":"CANTONMENT","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"2","Cust\_Name":"Sherin\_Selsia","Cust\_Phone\_No":9632145870,"Cust\_Gender":"Female","Cust\_DOB":"09/07/1970","Cust\_Acc\_Details":{"Acc\_No":1596789425,"Acc\_Type":"Current\_Account","Acc\_Balance":44552007,"Bank\_Info":{"IFSC\_Code":"SBIN00923","Branch\_Name":"CANTONMENT","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"3","Cust\_Name":"Chris\_Robert","Cust\_Phone\_No":8965471230,"Cust\_Gender":"Male","Cust\_DOB":"04/02/1986","Cust\_Acc\_Details":{"Acc\_No":4478945574,"Acc\_Type":"Joint\_Account","Acc\_Balance":15466636,"Bank\_Info":{"IFSC\_Code":"SBIN00923","Branch\_Name":"CANTONMENT","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"4","Cust\_Name":"Evan\_Sweety","Cust\_Phone\_No":8795642301,"Cust\_Gender":"Female","Cust\_DOB":"05/03/1988","Cust\_Acc\_Details":{"Acc\_No":4478945574,"Acc\_Type":"Joint\_Account","Acc\_Balance":15466636,"Bank\_Info":{"IFSC\_Code":"SBIN00923","Branch\_Name":"CANTONMENT","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"5","Cust\_Name":"Misheal\_Rithsia","Cust\_Phone\_No":8032146579,"Cust\_Gender":"Female","Cust\_DOB":"11/11/2002","Cust\_Acc\_Details":{"Acc\_No":2563256322,"Acc\_Type":"Savings\_Account","Acc\_Balance":3998520,"Bank\_Info":{"IFSC\_Code":"SBIN00923","Branch\_Name":"CANTONMENT","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"6","Cust\_Name":"Angelin\_Sweety","Cust\_Phone\_No":7560148975,"Cust\_Gender":"Female","Cust\_DOB":"12/06/1961","Cust\_Acc\_Details":{"Acc\_No":8755220545,"Acc\_Type":"Savings\_Account","Acc\_Balance":145830,"Bank\_Info":{"IFSC\_Code":"SBIN00923","Branch\_Name":"CANTONMENT","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"7","Cust\_Name":"Peter\_Pan","Cust\_Phone\_No":7032158463,"Cust\_Gender":"Male","Cust\_DOB":"12/03/1986","Cust\_Acc\_Details":{"Acc\_No":6655874236,"Acc\_Type":"Current\_Account","Acc\_Balance":5200354,"Bank\_Info":{"IFSC\_Code":"SBIY28791","Branch\_Name":"TVS\_TOLGATE","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"8","Cust\_Name":"Lovelin\_Jael","Cust\_Phone\_No":6540123547,"Cust\_Gender":"Female","Cust\_DOB":"04/11/1987","Cust\_Acc\_Details":{"Acc\_No":3248795632,"Acc\_Type":"Savings\_Account","Acc\_Balance":1545633,"Bank\_Info":{"IFSC\_Code":"SBIY28791","Branch\_Name":"TVS\_TOLGATE","Bank\_Name":"SBI"}},"Cust\_Loan\_Details":{"Loan\_ID":101,"Loan\_Type":"Educational\_Loan","Loan\_Amount":60000,"Installment\_Details":{"Amount":12000,"Start\_Date":"01/01/2020","End\_Date":"01/05/2020"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"9","Cust\_Name":"Issac\_Newton","Cust\_Phone\_No":6554447812,"Cust\_Gender":"Male","Cust\_DOB":"12/12/1981","Cust\_Acc\_Details":{"Acc\_No":3578956210,"Acc\_Type":"Current\_Account","Acc\_Balance":22323,"Bank\_Info":{"IFSC\_Code":"SBIY28791","Branch\_Name":"TVS\_TOLGATE","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"10","Cust\_Name":"Alice\_Dennis","Cust\_Phone\_No":6664442037,"Cust\_Gender":"Female","Cust\_DOB":"07/06/1996","Cust\_Acc\_Details":{"Acc\_No":2154512555,"Acc\_Type":"Savings\_Account","Acc\_Balance":22248832,"Bank\_Info":{"IFSC\_Code":"SBIY28791","Branch\_Name":"TVS\_TOLGATE","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"11","Cust\_Name":"Jennifer\_Susanna","Cust\_Phone\_No":9994442153,"Cust\_Gender":"Female","Cust\_DOB":"07/09/1976","Cust\_Acc\_Details":{"Acc\_No":6623236632,"Acc\_Type":"Savings\_Account","Acc\_Balance":2563222,"Bank\_Info":{"IFSC\_Code":"SBIA46780","Branch\_Name":"CRAWFORD","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"12","Cust\_Name":"Samuel\_Devakumar","Cust\_Phone\_No":9988774466,"Cust\_Gender":"Male","Cust\_DOB":"08/10/1961","Cust\_Acc\_Details":{"Acc\_No":2255897422,"Acc\_Type":"Savings\_Account","Acc\_Balance":255878,"Bank\_Info":{"IFSC\_Code":"SBIA46780","Branch\_Name":"CRAWFORD","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"13","Cust\_Name":"Fransina\_Protus","Cust\_Phone\_No":7782620855,"Cust\_Gender":"Female","Cust\_DOB":"09/01/1957","Cust\_Acc\_Details":{"Acc\_No":1566323381,"Acc\_Type":"Joint\_Account","Acc\_Balance":222233,"Bank\_Info":{"IFSC\_Code":"SBIN78291","Branch\_Name":"TENNUR","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"14","Cust\_Name":"Joseph\_Protus","Cust\_Phone\_No":6322015423,"Cust\_Gender":"Male","Cust\_DOB":"05/12/1950","Cust\_Acc\_Details":{"Acc\_No":1566323381,"Acc\_Type":"Joint\_Account","Acc\_Balance":222233,"Bank\_Info":{"IFSC\_Code":"SBIN78291","Branch\_Name":"TENNUR","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"15","Cust\_Name":"Clitus\_Joseph","Cust\_Phone\_No":7895461230,"Cust\_Gender":"Male","Cust\_DOB":"07/01/1946","Cust\_Acc\_Details":{"Acc\_No":5625632561,"Acc\_Type":"Current\_Account","Acc\_Balance":262332,"Bank\_Info":{"IFSC\_Code":"SBIN78291","Branch\_Name":"TENNUR","Bank\_Name":"SBI"}},"Cust\_Loan\_Details":{"Loan\_ID":102,"Loan\_Type":"Business\_Loan","Loan\_Amount":100000,"Installment\_Details":{"Amount":20000,"Start\_Date":"03/03/2020","End\_Date":"02/07/2020"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"16","Cust\_Name":"Frank\_Riverra","Cust\_Phone\_No":8794601235,"Cust\_Gender":"Male","Cust\_DOB":"08/12/1958","Cust\_Acc\_Details":{"Acc\_No":3663247415,"Acc\_Type":"Current\_Account","Acc\_Balance":781333,"Bank\_Info":{"IFSC\_Code":"SBIN78291","Branch\_Name":"TENNUR","Bank\_Name":"SBI"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"17","Cust\_Name":"Zaheer\_Husain","Cust\_Phone\_No":9845621754,"Cust\_Gender":"Male","Cust\_DOB":"03/12/1984","Cust\_Acc\_Details":{"Acc\_No":1236547890,"Acc\_Type":"Joint\_Account","Acc\_Balance":72300,"Bank\_Info":{"IFSC\_Code":"SBIN78291","Branch\_Name":"TENNUR","Bank\_Name":"SBI"}},"Cust\_Loan\_Details":{"Loan\_ID":103,"Loan\_Type":"Gold\_Loan","Loan\_Amount":150000,"Installment\_Details":{"Amount":50000,"Start\_Date":"01/09/2020","End\_Date":"1/11/2020"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"18","Cust\_Name":"Zainab\_Banu","Cust\_Phone\_No":8879954601,"Cust\_Gender":"Female","Cust\_DOB":"01/01/1986","Cust\_Acc\_Details":{"Acc\_No":1236547890,"Acc\_Type":"Joint\_Account","Acc\_Balance":72300,"Bank\_Info":{"IFSC\_Code":"SBIN78291","Branch\_Name":"TENNUR","Bank\_Name":"SBI"}},"Cust\_Loan\_Details":{"Loan\_ID":103,"Loan\_Type":"Gold\_Loan","Loan\_Amount":150000,"Installment\_Details":{"Amount":50000,"Start\_Date":"01/09/2020","End\_Date":"1/11/2020"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"19","Cust\_Name":"Simson\_Hermon","Cust\_Phone\_No":8975423015,"Cust\_Gender":"Male","Cust\_DOB":"02/06/1989","Cust\_Acc\_Details":{"Acc\_No":6321478956,"Acc\_Type":"Savings\_Account","Acc\_Balance":250000,"Bank\_Info":{"IFSC\_Code":"IOBA00193","Branch\_Name":"CANTONMENT","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"20","Cust\_Name":"Vimal\_Raj","Cust\_Phone\_No":9852078365,"Cust\_Gender":"Male","Cust\_DOB":"11/03/1979","Cust\_Acc\_Details":{"Acc\_No":6745218635,"Acc\_Type":"Current\_Account","Acc\_Balance":4662222,"Bank\_Info":{"IFSC\_Code":"IOBA00193","Branch\_Name":"CANTONMENT","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"21","Cust\_Name":"Juliet\_Mary","Cust\_Phone\_No":6545789721,"Cust\_Gender":"Female","Cust\_DOB":"12/08/1980","Cust\_Acc\_Details":{"Acc\_No":2663336633,"Acc\_Type":"Savings\_Account","Acc\_Balance":2623322,"Bank\_Info":{"IFSC\_Code":"IOBS780091","Branch\_Name":"CRAWFORD","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"22","Cust\_Name":"Jesudoss\_Jaikumar","Cust\_Phone\_No":6987745231,"Cust\_Gender":"Male","Cust\_DOB":"03/06/1974","Cust\_Acc\_Details":{"Acc\_No":5662234336,"Acc\_Type":"Savings\_Account","Acc\_Balance":2000786,"Bank\_Info":{"IFSC\_Code":"IOBS780091","Branch\_Name":"CRAWFORD","Bank\_Name":"IOB"}},"Cust\_Loan\_Details":{"Loan\_ID":201,"Loan\_Type":"Personal\_Loan","Loan\_Amount":400000,"Installment\_Details":{"Amount":40000,"Start\_Date":"02/01/2020","End\_Date":"01/10/2020"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"23","Cust\_Name":"Praveen\_Kumar","Cust\_Phone\_No":7895627013,"Cust\_Gender":"Male","Cust\_DOB":"04/12/1980","Cust\_Acc\_Details":{"Acc\_No":4789563210,"Acc\_Type":"Joint\_Account","Acc\_Balance":24578900,"Bank\_Info":{"IFSC\_Code":"IOBS780091","Branch\_Name":"CRAWFORD","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"24","Cust\_Name":"Priya\_Gannesh","Cust\_Phone\_No":7895462102,"Cust\_Gender":"Female","Cust\_DOB":"02/01/1981","Cust\_Acc\_Details":{"Acc\_No":4789563210,"Acc\_Type":"Joint\_Account","Acc\_Balance":24578900,"Bank\_Info":{"IFSC\_Code":"IOBS780091","Branch\_Name":"CRAWFORD","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"25","Cust\_Name":"Juliet\_Mary","Cust\_Phone\_No":8856472103,"Cust\_Gender":"Female","Cust\_DOB":"06/04/1984","Cust\_Acc\_Details":{"Acc\_No":5622311789,"Acc\_Type":"Savings\_Account","Acc\_Balance":5012477,"Bank\_Info":{"IFSC\_Code":"IOBD00456","Branch\_Name":"T\_NAGAR","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"26","Cust\_Name":"Anasar\_Amir","Cust\_Phone\_No":7546232254,"Cust\_Gender":"Male","Cust\_DOB":"07/12/1976","Cust\_Acc\_Details":{"Acc\_No":4589756123,"Acc\_Type":"Savings\_Account","Acc\_Balance":6013879,"Bank\_Info":{"IFSC\_Code":"IOBD00456","Branch\_Name":"T\_NAGAR","Bank\_Name":"IOB"}},"Cust\_Loan\_Details":{"Loan\_ID":202,"Loan\_Type":"Home\_Loan","Loan\_Amount":500000,"Installment\_Details":{"Amount":20000,"Start\_Date":"06/06/2019","End\_Date":"06/06/2021"}}})

WriteResult({ "nInserted" : 1 })

>

db.Customer.insert({\_id:"27","Cust\_Name":"Sundar\_Prakash","Cust\_Phone\_No":8974562300,"Cust\_Gender":"Male","Cust\_DOB":"12/13/1961","Cust\_Acc\_Details":{"Acc\_No":125879634,"Acc\_Type":"Joint\_Account","Acc\_Balance":25622005,"Bank\_Info":{"IFSC\_Code":"IOBD00456","Branch\_Name":"T\_NAGAR","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"28","Cust\_Name":"Shanthi\_Prakash","Cust\_Phone\_No":9443117956,"Cust\_Gender":"Female","Cust\_DOB":"12/12/1964","Cust\_Acc\_Details":{"Acc\_No":125879634,"Acc\_Type":"Joint\_Account","Acc\_Balance":25622005,"Bank\_Info":{"IFSC\_Code":"IOBD00456","Branch\_Name":"T\_NAGAR","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"29","Cust\_Name":"Ranjith\_Patel","Cust\_Phone\_No":9441263456,"Cust\_Gender":"Male","Cust\_DOB":"07/06/1994","Cust\_Acc\_Details":{"Acc\_No":2365853223,"Acc\_Type":"Current\_Account","Acc\_Balance":2552220,"Bank\_Info":{"IFSC\_Code":"IOBD00456","Branch\_Name":"T\_NAGAR","Bank\_Name":"IOB"}}})

WriteResult({ "nInserted" : 1 })

> db.Customer.insert({\_id:"30","Cust\_Name":"Suresh\_Kumar","Cust\_Phone\_No":8903242101,"Cust\_Gender":"Male","Cust\_DOB":"03/03/1988","Cust\_Acc\_Details":{"Acc\_No":2478963356,"Acc\_Type":"Savings\_Account","Acc\_Balance":45000217,"Bank\_Info":{"IFSC\_Code":"UBIN00123","Branch\_Name":"CANTONMENT","Bank\_Name":"UBI"}}})

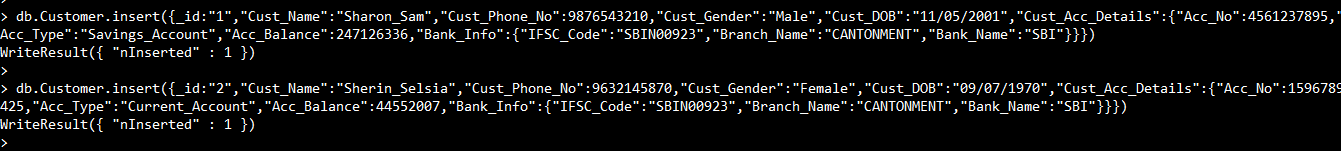
WriteResult({ "nInserted" : 1 })

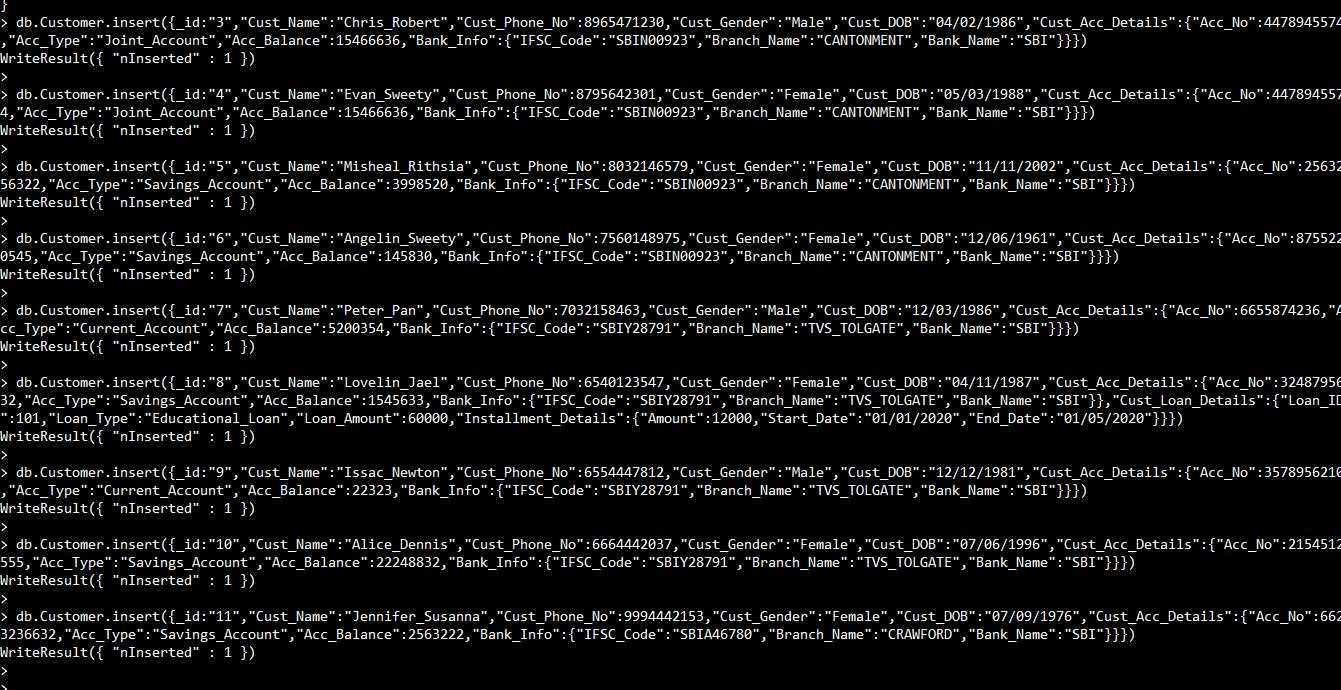
> db.Customer.insert({\_id:"31","Cust\_Name":"Simson\_Hermon","Cust\_Phone\_No":8975423015,"Cust\_Gender":"Male","Cust\_DOB":"02/06/1989","Cust\_Acc\_Details":{"Acc\_No":4587522652,"Acc\_Type":"Savings\_Account","Acc\_Balance":750000,"Bank\_Info":{"IFSC\_Code":"UBIN00123","Branch\_Name":"CANTONMENT","Bank\_Name":"UBI"}}})

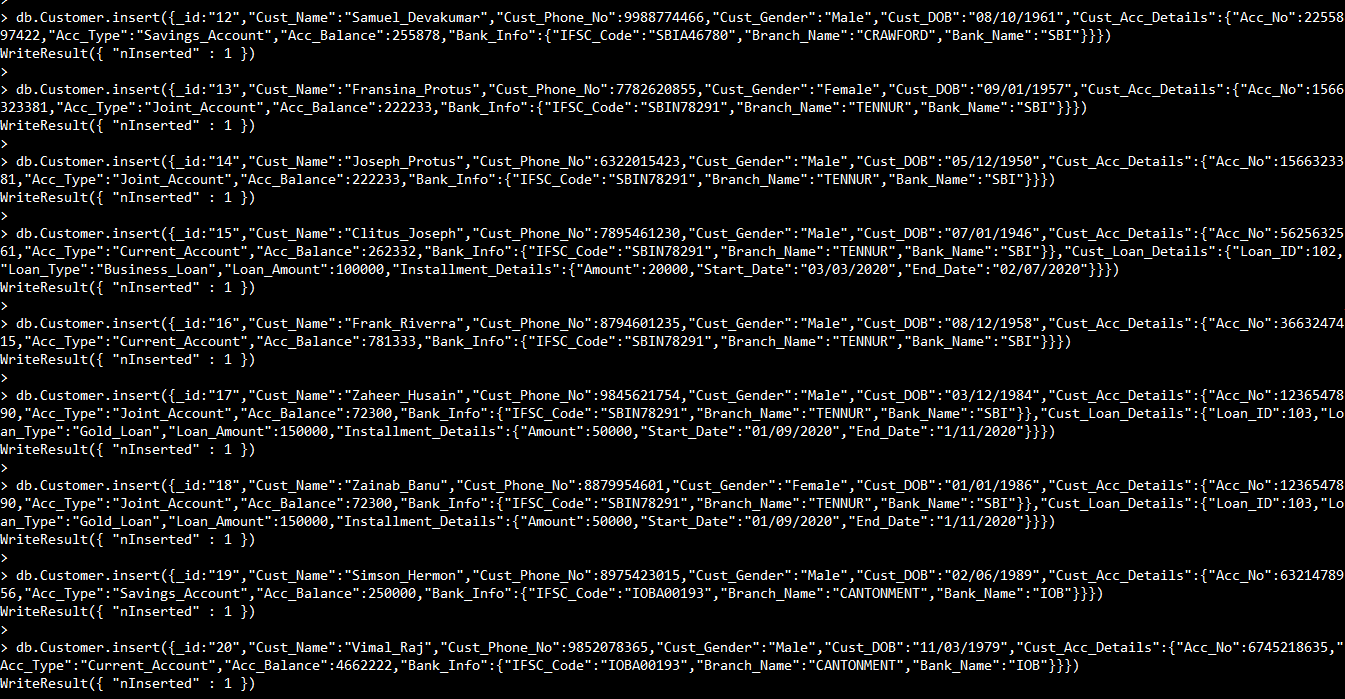
WriteResult({ "nInserted" : 1 })

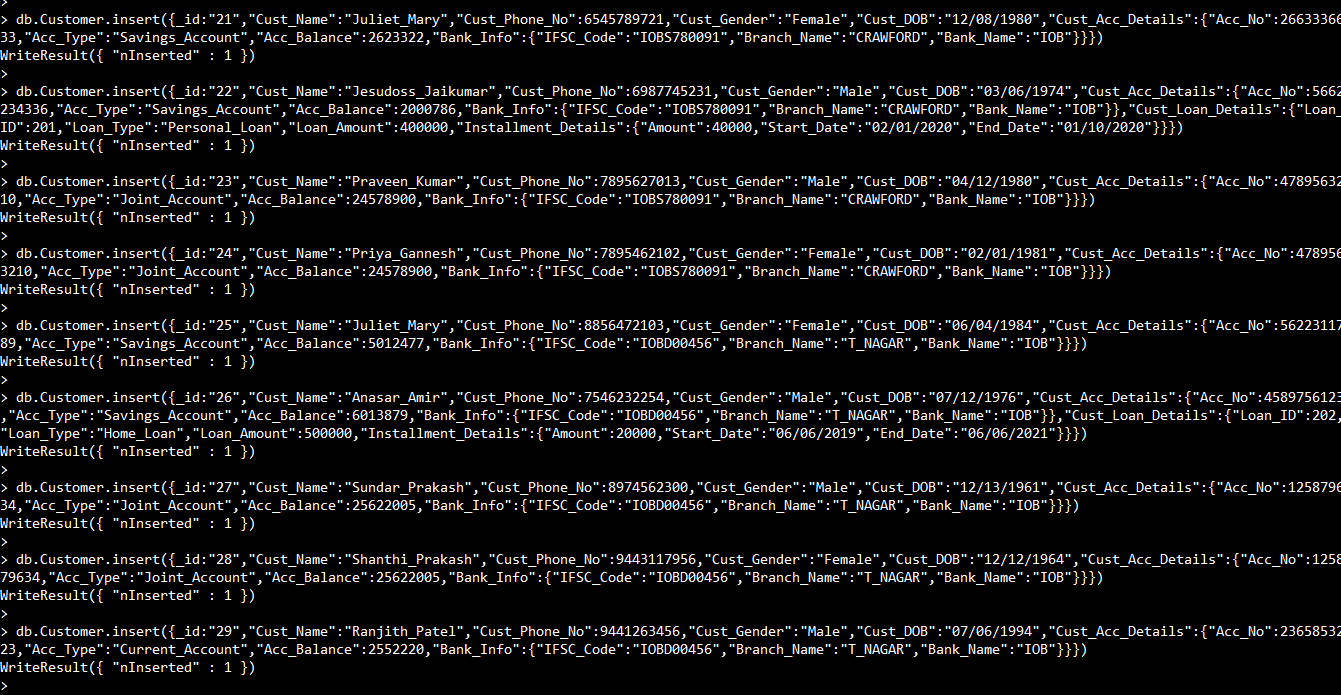
> db.Customer.insert({\_id:"32","Cust\_Name":"Yogesh","Cust\_Phone\_No":6265879320,"Cust\_Gender":"Male","Cust\_DOB":"05/04/2000","Cust\_Acc\_Details":{"Acc\_No":3644795562,"Acc\_Type":"Current\_Account","Acc\_Balance":457800,"Bank\_Info":{"IFSC\_Code":"UBIN00123","Branch\_Name":"CANTONMENT","Bank\_Name":"UBI"}}})

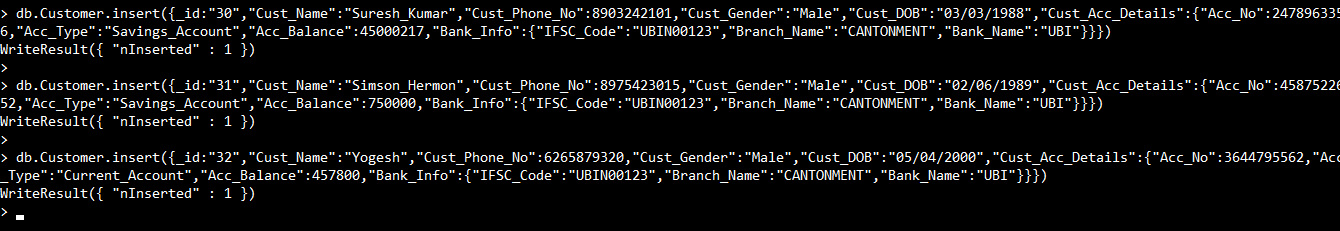
WriteResult({ "nInserted" : 1 })











**Updating the documents:**

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"SBIN00923"},{$set:{"Branch\_Address":"No.90,Main Road,Cantonment,Trichy-01"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"SBIY28791"},{$set:{"Branch\_Address":"No.30,Main Road,TVS Tolgate,Trichy-20"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"SBIA46780"},{$set:{"Branch\_Address":"No.2,Colony Road,CRAWFORD,Trichy-12"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"SBIN78291"},{$set:{"Branch\_Address":"No.2,K Road,TENNUR,Trichy-17"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"IOBA00193"},{$set:{"Branch\_Address":"No.20,Main Road,Cantonment,Trichy-01"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"IOBS780091"},{$set:{"Branch\_Address":"No.19,Anbu Nagar,CRAWFORD,Trichy-12"}});

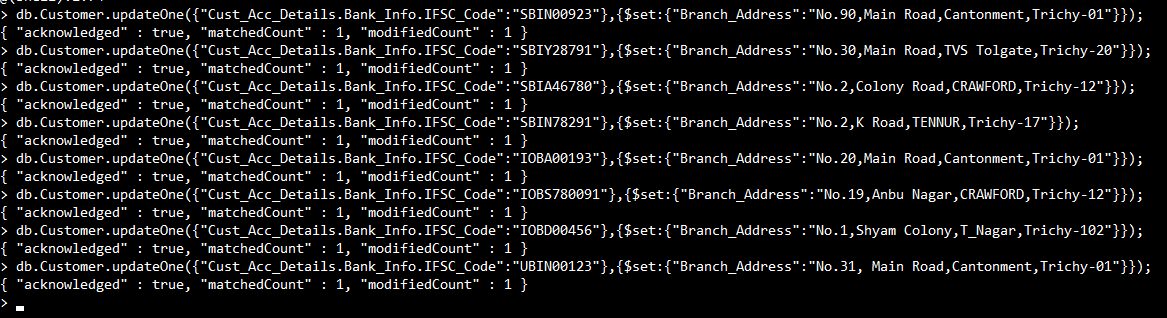
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"IOBD00456"},{$set:{"Branch\_Address":"No.1,Shyam Colony,T\_Nagar,Trichy-102"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.Customer.updateOne({"Cust\_Acc\_Details.Bank\_Info.IFSC\_Code":"UBIN00123"},{$set:{"Branch\_Address":"No.31, Main Road,Cantonment,Trichy-01"}});

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }



> db.Customer.update({"\_id":"1"},{$unset:{"Branch\_Address":1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Customer.update({"\_id":"7"},{$unset:{"Branch\_Address":1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Customer.update({"\_id":"11"},{$unset:{"Branch\_Address":1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Customer.update({"\_id":"13"},{$unset:{"Branch\_Address":1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Customer.update({"\_id":"19"},{$unset:{"Branch\_Address":1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Customer.update({"\_id":"21"},{$unset:{"Branch\_Address":1}})

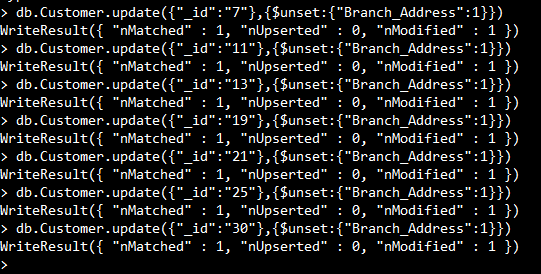
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Customer.update({"\_id":"25"},{$unset:{"Branch\_Address":1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.Customer.update({"\_id":"30"},{$unset:{"Branch\_Address":1}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })



**Selecting the documents:**

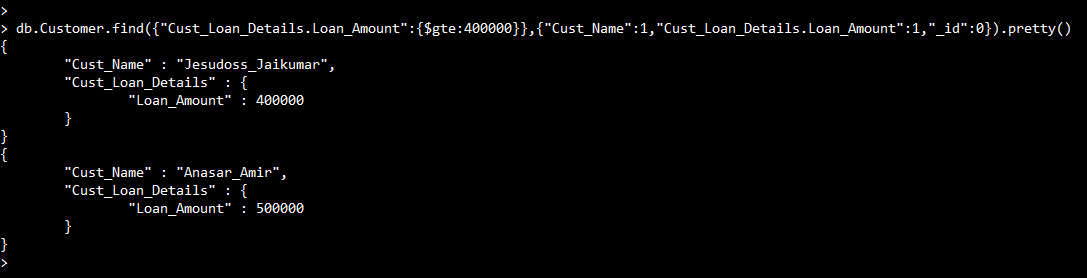
> db.Customer.find().pretty()



**Comparison Operations:**

**Finding the customer name with the loan amount who borrowed from four lakhs**

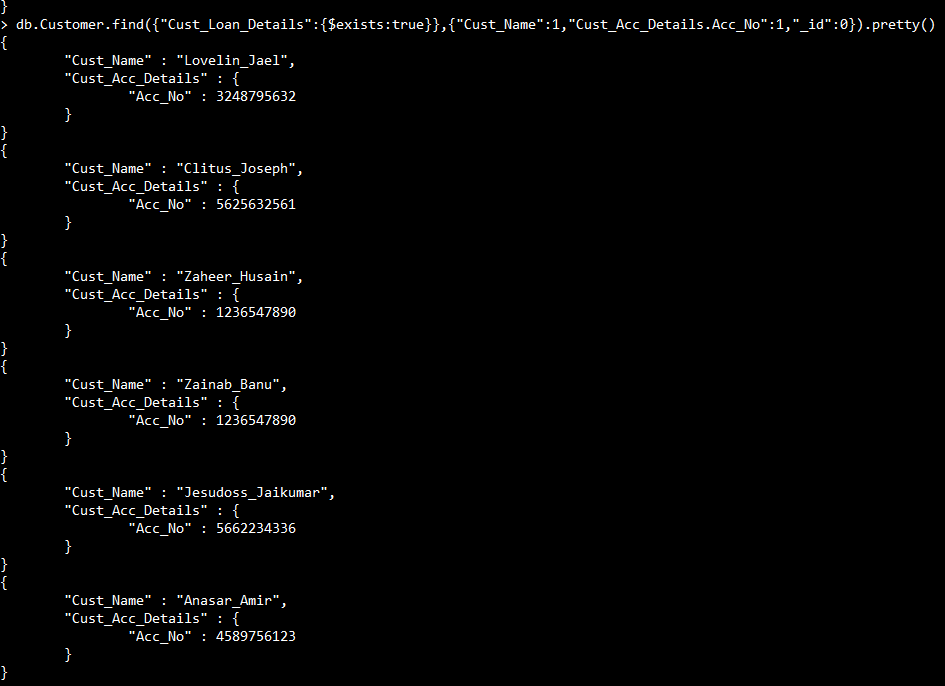
> db.Customer.find({"Cust\_Loan\_Details.Loan\_Amount":{$gte:400000}},{"Cust\_Name":1,"Cust\_Loan\_Details.Loan\_Amount":1,"\_id":0}).pretty()



**Logical Operations:**

**Finding the customer name with account number for those who availed the loan from bank**

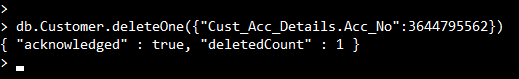
> db.Customer.find({"Cust\_Loan\_Details":{$exists:true}},{"Cust\_Name":1,"Cust\_Acc\_Details.Acc\_No":1,"\_id":0}).pretty()



**Deleting the Documents:**

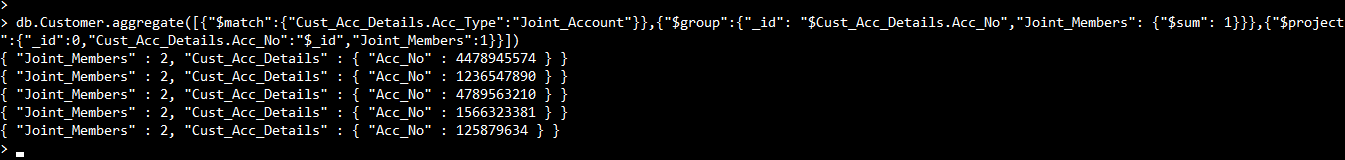
> db.Customer.deleteOne({"Cust\_Acc\_Details.Acc\_No":3644795562})

{ "acknowledged" : true, "deletedCount" : 1 }



**Aggregation:**

> db.Customer.aggregate([{"$match":{"Cust\_Acc\_Details.Acc\_Type":"Joint\_Account"}},{"$group":{"\_id": "$Cust\_Acc\_Details.Acc\_No","Joint\_Members": {"$sum": 1}}},{"$project":{"\_id":0,"Cust\_Acc\_Details.Acc\_No":"$\_id","Joint\_Members":1}}])

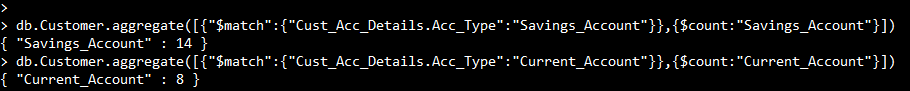


> db.Customer.aggregate([{"$match":{"Cust\_Acc\_Details.Acc\_Type":"Savings\_Account"}},{$count:"Savings\_Account"}])

{ "Savings\_Account" : 14 }

> db.Customer.aggregate([{"$match":{"Cust\_Acc\_Details.Acc\_Type":"Current\_Account"}},{$count:"Current\_Account"}])

{ "Current\_Account" : 8 }



**Graph Database:**

A graph database stores nodes and relationships instead of tables, or documents. Your data is stored without restricting it to a pre-defined model, allowing a very flexible way of thinking about and using it.

Modern Banking Products are fueled by a User-Centric approach and currentily Banks wanted to create a modern banking experience that broke with traditional account-centric banking. They use the Neo4j Graph Database as a flexible framework for building innovative banking products that leverage relationships. The main usage is for being able to quickly traverse to different data elements, and are able to encode the data model as a primitive directly into the database.

**Neo4j:**

Neo4j is an open-source graph database and also called as Property Graph Model. In Neo4j, information is organized as nodes, relationships, and properties.

The building blocks of the property graph model are the Nodes and Relationships where Nodes are the entities in the graph which can be tagged with labels with some properties and relationships provide directed, named, connections between two node entities. Nodes can have any number or type of relationships without sacrificing performance.

**CQL (Cypher Query Language):**

Cypher is a declarative query language optimized for graphs.

Cypher is Neo4j’s graph query language that lets you retrieve data from the graph. Cypher is unique because it provides a visual way of matching patterns and relationships. Neo4j users use Cypher to construct expressive and efficient queries where Cypher being the primary interface for Neo4j.

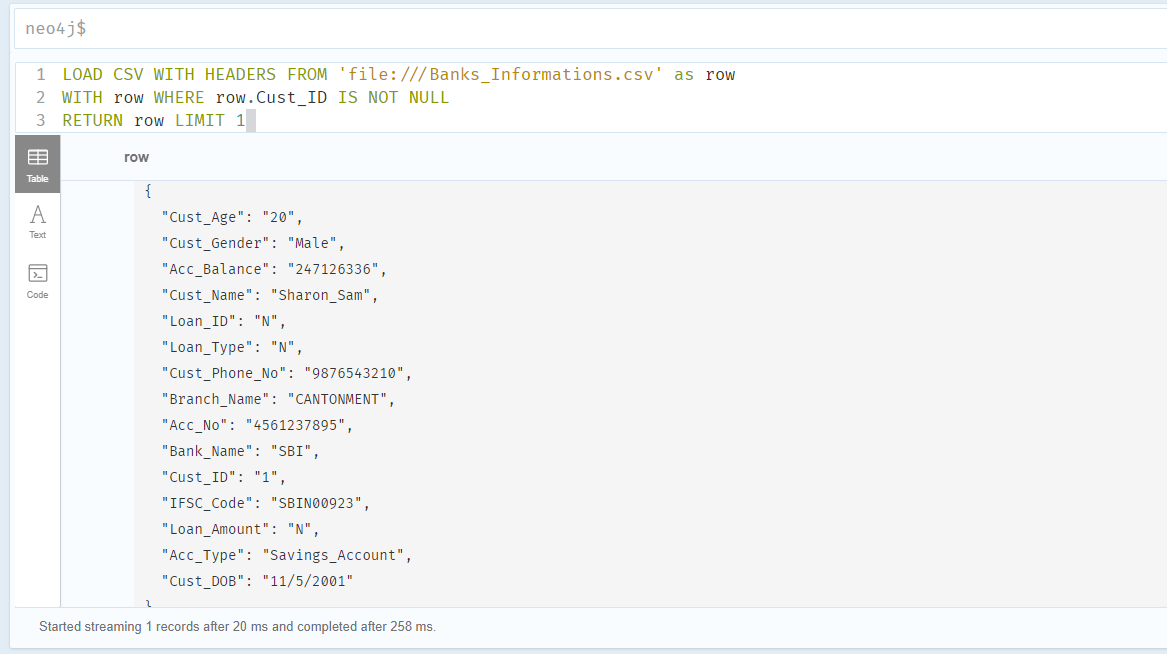
**CQL Queries:**

**Importing the csv files into the database and Reading the entries**

LOAD CSV WITH HEADERS FROM 'file:///Banks\_Informations.csv' as row

WITH row WHERE row.Cust\_ID IS NOT NULL

RETURN row LIMIT 1

****

**Creating Nodes and Relationships**

LOAD CSV WITH HEADERS FROM 'file:///Banks\_Informations.csv' as row

WITH row WHERE row.Cust\_ID IS NOT NULL

MERGE (B:BANK {Bank\_Name:row.Bank\_Name})

MERGE (BR:BRANCH {IFSC\_Code:row.IFSC\_Code, Branch\_Name:row.Branch\_Name})

MERGE (A:ACCOUNT {Acc\_No:row.Acc\_No, Acc\_Type:row.Acc\_Type, Acc\_Balance:toInteger(row.Acc\_Balance)})

MERGE (L:LOAN {Loan\_ID:row.Loan\_ID, Loan\_Type:row.Loan\_Type, Loan\_Amount:row.Loan\_Amount})

MERGE (C:CUSTOMER {Cust\_ID:row.Cust\_ID, Cust\_Name:row.Cust\_Name, Cust\_Phone\_No:row.Cust\_Phone\_No, Cust\_Gender:row.Cust\_Gender, Cust\_DOB:row.Cust\_DOB, Cust\_Age:row.Cust\_Age})

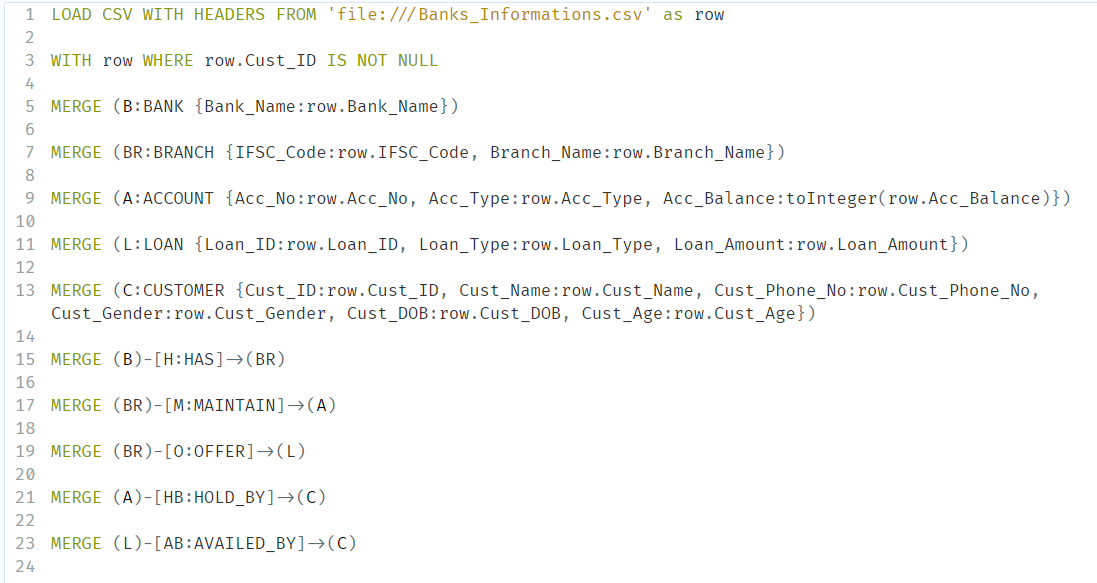
MERGE (B)-[H:HAS]->(BR)

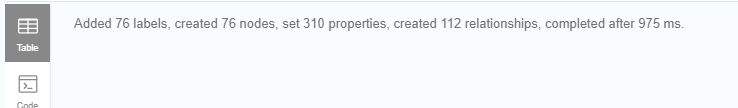
MERGE (BR)-[M:MAINTAIN]->(A)

MERGE (BR)-[O:OFFER]->(L)

MERGE (A)-[HB:HOLD\_BY]->(C)

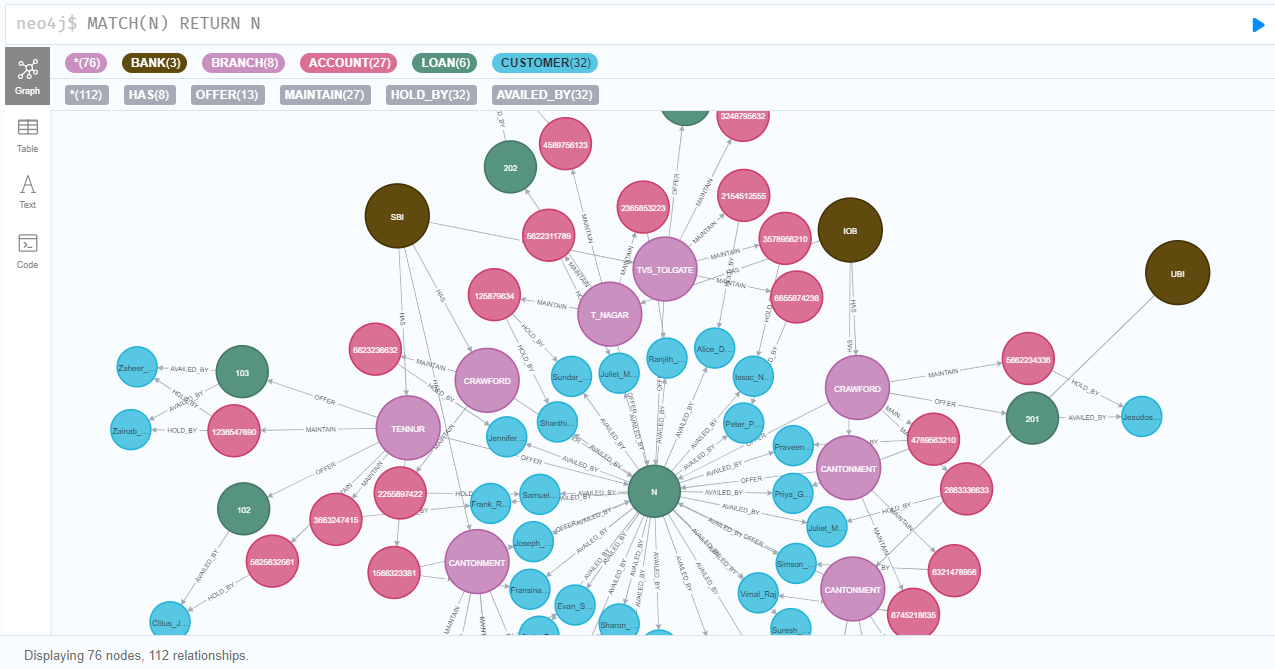
MERGE (L)-[AB:AVAILED\_BY]->(C)





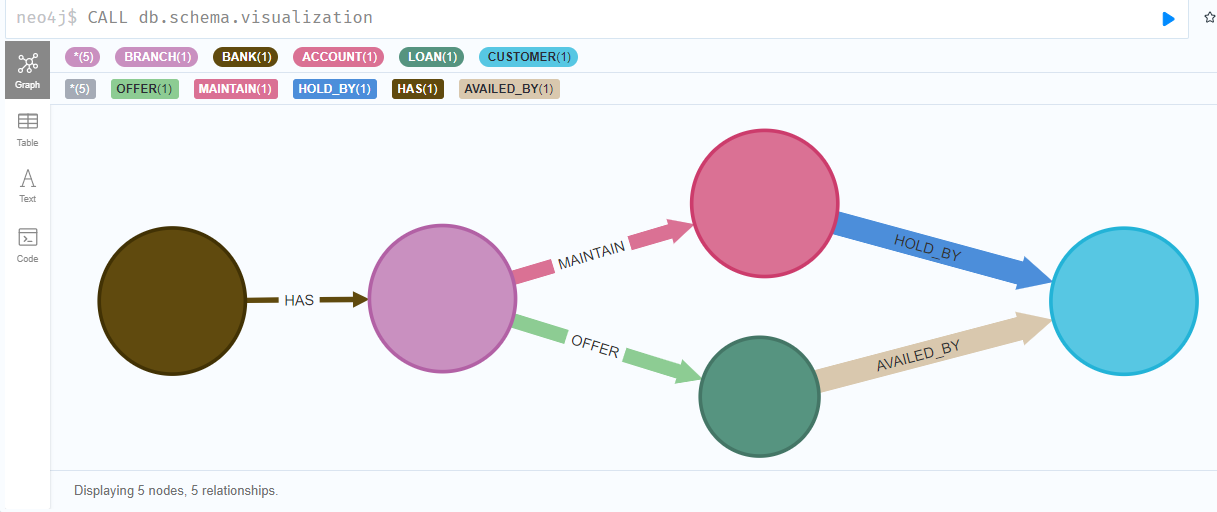
**Retriving all nodes from the database**

MATCH(N) RETURN N



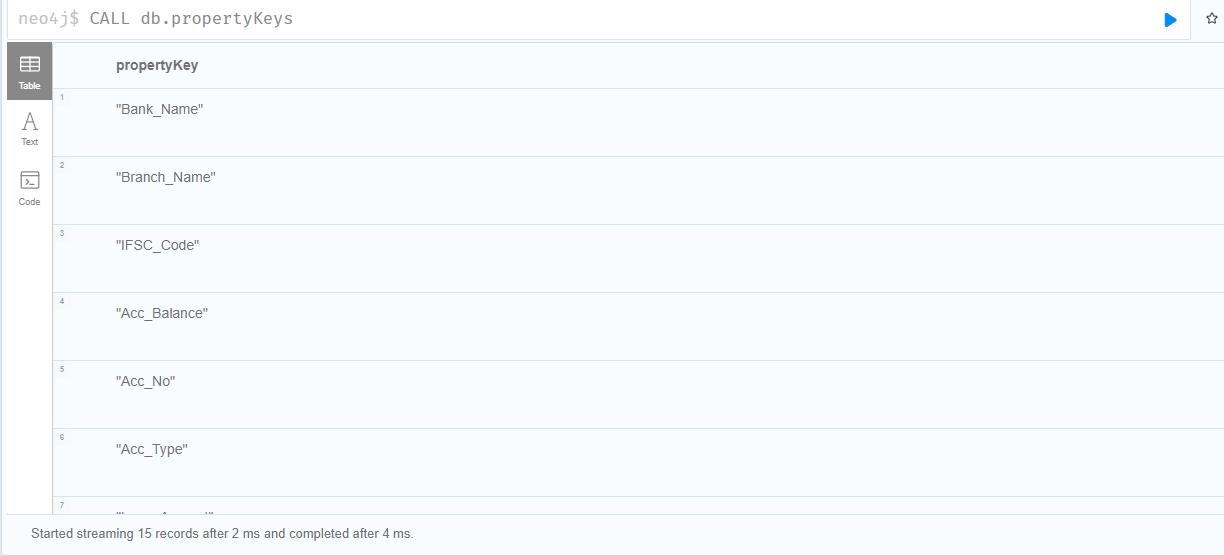
**Showing the schema of the database**

CALL db.schema.visualization



**Showing all the property keys in the database**

CALL db.propertyKeys



**Conclusion & Confrontations:**

* Considering the constraints for modeling a DBMS using the E-R diagram.
* Long tedious process of creating excel files for importing in Oracle environment.
* Inserting documents in MongoDB environment.
* Creating relationships for the nodes in Neo4j.

The scope of understanding the case study for BANKING ENTERPRISE have been grasped and developed the knowledge by designing the database using E-R Diagram and creating a database for maintaining the records in Oracle by developing SQL queries; MongoDB by developing NoSQL queries; Neo4j by developing CQL queries.

MONITOR the MONEY with DBMS……