***Import Statements***

**Creating Nodes**

Account Nodes:

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///account.csv" AS row

CREATE (:Account {accountNum: toInteger(row.AcctNumber), openDate: date(row.OpenDate), accountStatus: row.AcctStatus, statusDate: date(row.StatusDate), closeDate: date(row.CloseDate), currentBalance: toFloat(row.CurrentBalance), productCode: row.ProductCode});

Individual and Company Customer Nodes:

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///individual\_customer.csv" AS row

CREATE (:Customer{customerID: toInteger(row.Cust\_ID), customerType: row.Cust\_Type, taxID: row.Tax\_ID, firstName: row.FirstName, lastName: row.LastName, middleName: row.MiddleName, birthDate: row.BirthDate, primaryCitizenship: row.Primary\_Citizenship, countryRes: row.CountryRes, employerName: row.EmployerName, employmentStatus: row.EmployStatus, maritalStatus: row.MaritalStatus, occupation: row.Occupation, orgName: row.OrgName, employeeIND: row.EmployeeIND, customerStatus: row.CustomerStatus, address1: row.Address1, address2: row.Address2, address3: row.Address3, city: row.City, state: row.State, zipcode: row.Zipcode, country: row.Country});

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///company\_customer.csv" AS row

CREATE (:Customer{customerID: toInteger(row.Cust\_ID), customerType: row.Cust\_Type, taxID: row.Tax\_ID, firstName: row.FirstName, lastName: row.LastName, middleName: row.MiddleName, birthDate: row.BirthDate, primaryCitizenship: row.Primary\_Citizenship, countryRes: row.CountryRes, employerName: row.EmployerName, employmentStatus: row.EmployStatus, maritalStatus: row.MaritalStatus, occupation: row.Occupation, orgName: row.OrgName, employeeIND: row.EmployeeIND, customerStatus: row.CustomerStatus, address1: row.Address1, address2: row.Address2, address3: row.Address3, city: row.City, state: row.State, zipcode: row.Zipcode, country: row.Country});

Transaction Nodes

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///transaction.csv" AS row

CREATE (:Transaction{tracewireID: toInteger(row.Trace\_Wire\_ID), tAccount1: toInteger(row.Account\_ID), tAccount2: toInteger(row.Account\_ID\_2), tAmount: toFloat(row.Transaction\_Amount), tDescription: row.Transaction\_Descrip, tCode: row.Transaction\_Code, DorC: row.DC});

Date Nodes

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///date.csv" AS row

CREATE (:Date{date: date(row.Date)});

**Creating Indexes (Primary Keys)**

CREATE INDEX ON :Account(accountNum);

CREATE INDEX ON :Customer(customerID);

**Creating Relationships**

Transaction 🡪 Date

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///transaction\_to\_date.csv" AS row

MATCH(transaction:Transaction {tracewireID: toInteger(row.Trace\_Wire\_ID)})

MATCH(date:Date {date: date(row.Transaction\_Date)})

MERGE(transaction)-[:TRANSACTION\_DATE]->(date);

Individuals 🡪 Accounts

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///customer\_to\_account\_individual.csv" AS row

MATCH(customer:Customer {customerID: toInteger(row.Customer\_ID)})

MATCH(account:Account {accountNum: toInteger(row.Account\_ID)})

MERGE(customer)-[:OWNS]->(account);

Companies 🡪 Accounts

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///customer\_to\_account\_company.csv" AS row

MATCH(customer:Customer {customerID: toInteger(row.Customer\_ID)})

MATCH(account:Account {accountNum: toInteger(row.Account\_ID)})

MERGE(customer)-[:OWNS]->(account);

Account1 🡪 Account2

MATCH(a1:Account),(t:Transaction),(a2:Account)

WHERE a1.accountNum = t.tAccount1 AND a2.accountNum = t.tAccount2

MERGE(a1)-[p:TRANSACTS\_WITH]->(a2);

Account1 -> Transaction

MATCH(a:Account),(t:Transaction)

WHERE a.accountNum = t.tAccount1

MERGE(a)-[:TRANSACTION\_IN]->(t);

Transaction 🡪 Account2

MATCH(a:Account),(t:Transaction)

WHERE a.accountNum = t.tAccount2

MERGE(t)-[:TRANSACTION\_OUT]->(a);

Extra Database Features:

Party to Account

USING PERIODIC COMMIT

LOAD CSV WITH HEADERS FROM "file:///OWNER\_OF.csv" AS row

MATCH(customer:Customer { customerID:toInteger(row.customerid)})

MATCH(account:Account {accountNum:toInteger(row.accountid)})

MERGE(customer)-[:OWNER\_OF]->(account)

Party to Party

USING PERIODIC COMMIT  
LOAD CSV WITH HEADERS FROM “file” AS row

MATCH(customer1:Customer { customerID:toInteger(row.customerid1)})

MATCH(customer2:Customer { customerID:toInteger(row.customerid2)})

MERGE(customer1)-[:RELATIONSHIP]->(customer2)