Neo4j imports

Account

```
CREATE CONSTRAINT FOR (a:Account) REQUIRE a.accountId IS UNIQUE;
```

```
Load csv with headers from 'file:///Account.csv' as row
CREATE
(a:Account {accountId: toInteger(row.accountId)}) SET a.createTime = datetime(row.createT
ime), a.isBlocked = toBoolean(row.isBlocked), a.accountType = row.accoutType, a.nickname
= row.nickname, a.phonenum = toString(row.phonenum), a.email = row.email,
a.freqLoginType = row.freqLoginType, a.lastLoginTime = datetime({epochmillis:
toInteger(row.lastLoginTime)}), a.accountLevel = row.accountLevel
```

Medium

```
CREATE CONSTRAINT FOR (m:Medium) REQUIRE m.mediumId IS UNIQUE;
```

```
Load csv with headers from 'file:///Medium.csv' as row
CREATE (m:Medium {mediumId: toInteger(row.mediumId)}) SET m.mediumType = row.m
ediumType, m.isBlocked = toBoolean(row.isBlocked), m.createTime = datetime(row.createTi
me), m.lastLoginTime = datetime({epochmillis: toInteger(row.lastLoginTime)}), m.riskLevel =
row.riskLevel
```

Person

```
CREATE CONSTRAINT FOR (p:Person) REQUIRE p.personId IS UNIQUE;
```

```
Load csv with headers from 'file:///Person.csv' as row
CREATE (p:Person {personId: toInteger(row.personId)}) SET p.personName = row.personN
ame, p.isBlocked = toBoolean(row.isBlocked), p.createTime = datetime(row.createTime),
p.gender = row.gender, p.birthday = date(row.birthday), p.country = row.country, p.city = row.city
```

Loan

```
CREATE CONSTRAINT FOR (l:Loan) REQUIRE l.loanId IS UNIQUE;
```

```
Load csv with headers from 'file:///Loan.csv' as row
CREATE (l:Loan {loanId: toInteger(row.loanId)}) SET l.loanAmount = toFloat(row.
loanAmount), l.balance = toFloat(row.balance), l.createTime = datetime(row.createTime),
l.loanUsage = row.loanUsage, l.interestRate = toFloat(row.interestRate)
```

Company

```
CREATE CONSTRAINT FOR (c:Company) REQUIRE c.companyId IS UNIQUE;
```

```
Load csv with headers from 'file:///Company.csv' as row
CREATE (c:Company {companyId: toInteger(row.companyId)}) SET c.companyName = ro
w. companyName, c.isBlocked
= toBoolean(row.isBlocked), c.createTime = datetime(row.createTime), c.country =
row.country, c.city = row.city, c.business = row.business, c.description =
c.url = row.url
```

AccountRepayLoan

```
Load csv with headers from 'file:///AccountRepayLoan.csv' as row
Match (a:Account{accountId: toInteger(row.accountId)})
Match (l:Loan{loanId: toInteger(row.loanId)})
CREATE (a)-[:repay {amount: toFloat(row.amount),
createTime: datetime(row.createTime)}]->(l)
```

AccountTransferAccount

```
Load csv with headers from 'file:///AccountTransferAccount.csv' as row
Match (a1:Account{accountId: toInteger(row.fromId)})
Match (a2:Account{accountId: toInteger(row.toId)})
CREATE (a1)-
[:transfer {amount: toFloat(row.amount), createTime: datetime(row.createTime), orderNum:
row.orderNum, comment: row.comment, payType: row.payType, goodsType:
row.goodsType}]->(a2)
```

AccountWithdrawAccount

```
Load csv with headers from 'file:///AccountWithdrawAccount.csv' as row
Match (a1:Account{accountId: toInteger(row.fromId)})
Match (a2:Account{accountId: toInteger(row.toId)})
CREATE (a1)-
[:withdraw {amount: toFloat(row.amount), createTime: datetime(row.createTime)}]->(a2)
```

CompanyApplyLoan

```
Load csv with headers from 'file:///CompanyApplyLoan.csv' as row
Match (c:Company{companyId: toInteger(row.companyId)})
Match (l:Loan{loanId: toInteger(row.loanId)})
CREATE (c)-[:apply {createTime: datetime(row.createTime), org: row.org}]->(l)
```

CompanyGuaranteeCompany

```
Load csv with headers from 'file:///CompanyGuaranteeCompany.csv' as row
Match (c1:Company{companyId: toInteger(row.fromId)})
Match (c2:Company{ companyId: toInteger(row. toId)})
CREATE (c1)-[:guarantee {createTime: datetime(row.createTime), relation: row.relation}]-
>(c2)
```

CompanyInvestCompany

```
Load csv with headers from 'file:///CompanyInvestCompany.csv' as row
Match (c1:Company{companyId: toInteger(row.investorId)})
Match (c2:Company{ companyId: toInteger(row. companyId)})
```

```
CREATE (c1)-[:invest {ratio: toFloat(row.ratio), createTime: datetime(row.createTime)}]->(c2)
```

CompanyOwnAccount

```
Load csv with headers from 'file:///CompanyOwnAccount.csv' as row
Match (c:Company{companyId: toInteger(row.companyId)})
Match (a:Account{accountId: toInteger(row.accountId)})
CREATE (c)-[:own {createTime: datetime(row.createTime)}]->(a)
```

LoanDepositAccount

```
Load csv with headers from 'file:///LoanDepositAccount.csv' as row
Match (l:Loan{loanId: toInteger(row.loanId)})
Match (a:Account{accountId: toInteger(row.accountId)})
CREATE (l)-[:deposit {amount: toFloat(row.amount),
createTime: datetime(row.createTime)}]->(a)
```

MediumSignInAccount

```
Load csv with headers from 'file:///MediumSignInAccount.csv' as row
Match (m:Medium{mediumId: toInteger(row.mediumId)})
Match (a:Account{accountId: toInteger(row.accountId)})
CREATE (m)-[:signIn {createTime: datetime(row.createTime), location: row.location}]->(a)
```

PersonApplyLoan

```
Load csv with headers from 'file:///PersonApplyLoan.csv' as row
Match (p:Person{personId: toInteger(row.personId)})
Match (l:Loan{loanId: toInteger(row.loanId)})
CREATE (p)-[:apply {createTime: datetime(row.createTime), org: row.org}]->(l)
```

PersonGuaranteePerson

```
Load csv with headers from 'file:///PersonGuaranteePerson.csv' as row
Match (p1:Person{personId: toInteger(row.fromId)})
Match (p2:Person{personId: toInteger(row.toId) })
CREATE (p1)-[:guarantee {createTime: datetime(row.createTime), relation: row.relation}]->(p2)
```

PersonInvestCompany

```
Load csv with headers from 'file:///PersonInvestCompany.csv' as row
Match (p:Person{personId: toInteger(row.investorId)})
Match (c:Company{companyId: toInteger(row.companyId)})
CREATE (p)-[:invest {ratio: toFloat(row.ratio), createTime: datetime(row.createTime)}]->(c)
```

PersonOwnAccount

```
Load csv with headers from 'file:///PersonOwnAccount.csv' as row
Match (p:Person{personId: toInteger(row.personId)})
Match (a:Account{accountId: toInteger(row.accountId)})
CREATE (p)-[:own {createTime: datetime(row.createTime)}]->(a)
```