Neo4j – CREATE NODES AND RELATIONSHIPS

fashion_designers:

```
LOAD CSV WITH HEADERS FROM 'file:///fashion_designers.csv' AS row
WITH row WHERE row.personal_id IS NOT NULL
MERGE (fd: fashion_designer {personal_id: toInteger(row.personal_id),
                    name: row.name,
                    surname: row.surname,
                    phone_number: row.phone_number,
                    stage_name: row.stage_name,
                    career: row.career
                   });
```

fashon_models:

```
LOAD CSV WITH HEADERS FROM 'file:///fashion_models.csv' AS row
WITH row WHERE row.personal_id IS NOT NULL
MERGE (fm: fashion_model {personal_id: toInteger(row.personal_id),
                    name: row.name,
                    surname: row.surname,
                    phone_number: row.phone_number,
                    height: toInteger(row.height),
                    size: row.size,
                    date_of_birth: row.date_of_birth
                   });
```

dresses:

```
LOAD CSV WITH HEADERS FROM 'file:///dresses.csv' AS row
WITH row WHERE row.id IS NOT NULL
MERGE (d: dress {id: toInteger(row.id),
        size: row.size,
        colour: row.colour,
        production_time: toFloat(row.production_time)
       });
```

fashion_exhibitions:

```
LOAD CSV WITH HEADERS FROM 'file:///fashion_exhibitions.csv' AS row
WITH row WHERE row.id IS NOT NULL
MERGE (fe: fashion_exhibition {id: toInteger(row.id),
                duration: toFloat(row.duration),
                location_address: row.location_address,
                title: row.title,
                description: row.description
               });
```

fabrics:

```
LOAD CSV WITH HEADERS FROM 'file:///fabrics.csv' AS row
WITH row WHERE row.id IS NOT NULL
MERGE (f: fabric {id: toInteger(row.id),
        price_per_square_meter: toFloat(row.price_per_square_meter),
        name: row.name
        });
```

Fabrics' attribute "properties":

```
LOAD CSV WITH HEADERS FROM 'file:///properties_of_fabrics.csv' AS row
MATCH (fa:fabric {id: toInteger(row.fabrics_id)})
WITH fa, collect(row.properties_name) AS props
SET fa.properties = props
```

Relationship between fashion_designers and fashion_models:

```
LOAD CSV WITH HEADERS FROM 'file:///fashion_models.csv' AS row
MATCH (fm: fashion_model {personal_id: toInteger(row.personal_id)})
MATCH (fd: fashion_designer {personal_id: toInteger(row.fashion_designers_personal_id)})
CREATE (fd) - [m: manages] -> (fm)
```

NOTE: By creating the relationship in this way, each designer will link their respective models.

Differently, if we had kept the relationship as implemented in the ER model, then we would have that each model connects the respective designer to which it is assigned, changing the direction of the relationship.

Relationship between fashion_models and dresses:

```
LOAD CSV WITH HEADERS FROM 'file:///dresses.csv' AS row
MATCH (d: dress {id: toInteger(row.id)})
MATCH (fm: fashion_model {personal_id: toInteger(row.fashion_models_personal_id)})
CREATE (fm) - [w: wears] -> (d)
```

Relationship between dresses and fabrics:

```
LOAD CSV WITH HEADERS FROM 'file:///composed_of.csv' AS row
MATCH (d: dress {id: toInteger(row.dresses_id)})
MATCH (f: fabric {id: toInteger(row.fabrics_id)})
CREATE (d) - [c: composed_of {amount: toInteger(row.amount)}] -> (f)
```

Relationship between fashion_models and fashion_exhibitions:

```
LOAD CSV WITH HEADERS FROM 'file:///partake_to.csv' AS row
MATCH (fe: fashion_exhibition {id: toInteger(row.fashion_exhibitions_id)})
MATCH (fm: fashion_model {personal_id: toInteger(row.fashion_models_personal_id)})
CREATE (fm) - [p: partake_to] -> (fe)
```