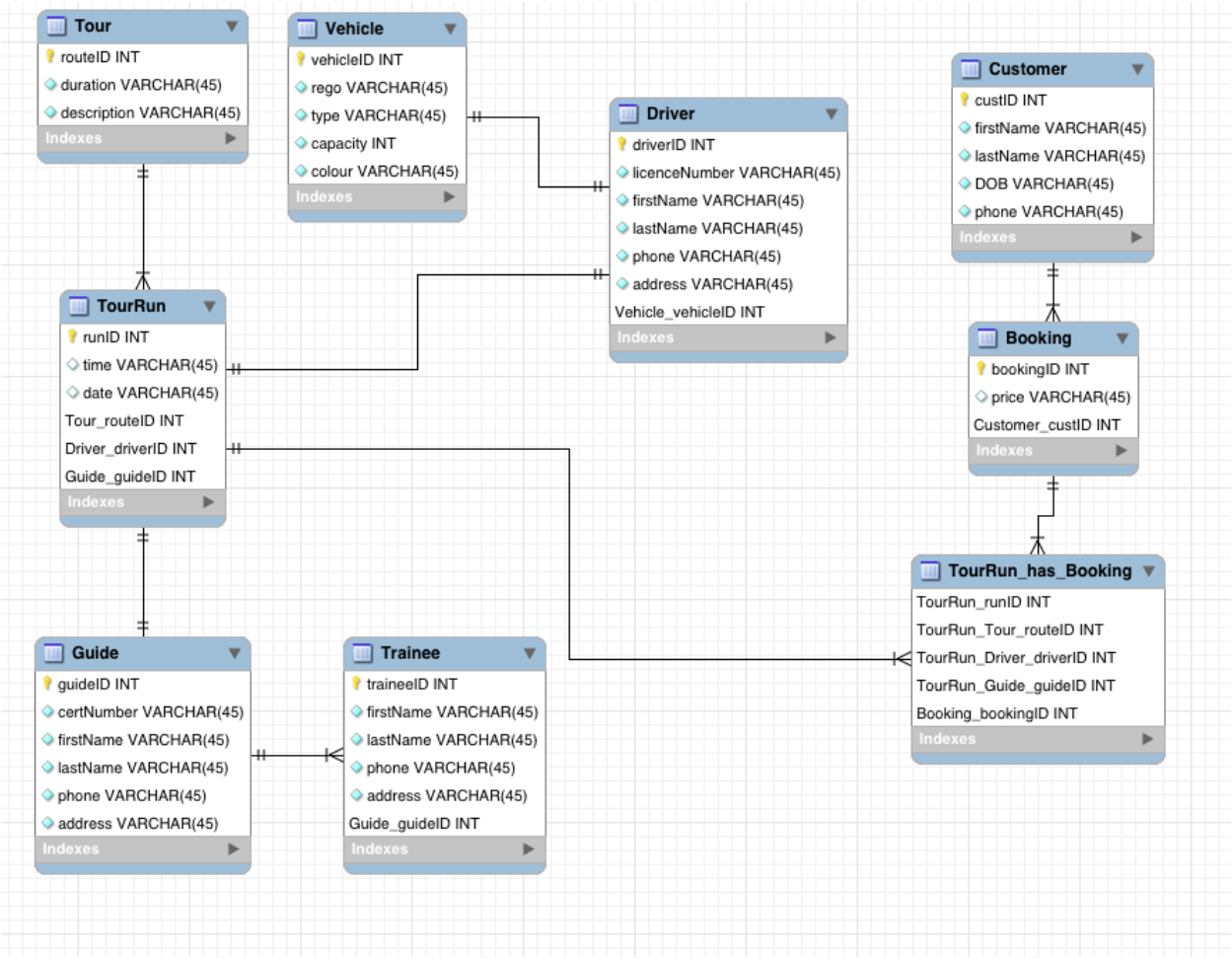
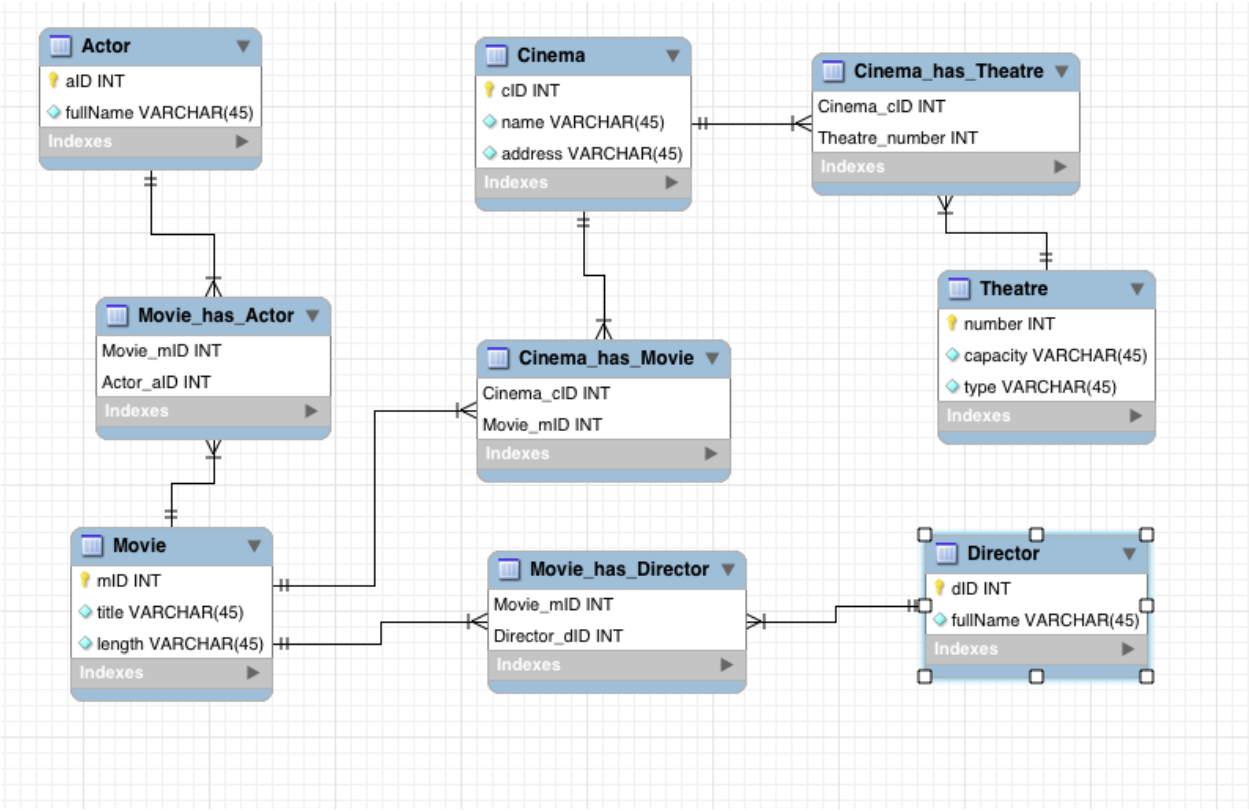
Question 1: ER Model



Question 2: Relational Model



Question 3: Relational Design

3.1.1.

3.1.2. F,H,I,J+

3.1.3.

3.2.1 StaffID, StoreAddress, SalaryGroup

3.2.2 staffID+

3.2.3

3.2.4

Question 4: Short Answer

Ternary relationships allow easer database expansion over complicated multi binary relationships, for the ER diagrams listed, the diagram on the left not only is simpler it does not double up on tables like the diagram on the right e.g. (Stud_Borrow / Staff_Borrow), not having double up tables will make the database preform faster and make troubleshooting easier should an issue arise.

Question 5: Advanced SQL

Write an SQL Query for each question below to extract information from the database

1. Which couple works at the same bank (same address)(no repeted pairs)

```
SELECT *
FROM employee
INNER JOIN (SELECT address
                FROM   employee
                GROUP  BY address
                HAVING COUNT(employeeID) > 1) dup
        ON employee.address = dup.address;
```

2. Using IN or NOT IN compose a query that can determine which employees are currently not working

```
Select employee.employeeID, employee.firstName, employee.lastName
From employee
Join worksAt On worksAt.employeeID = employee.employeeID
Where worksAt.occupation in
(Select occupation From worksAt Where occupation = null or occupation = '')
```

3. Find all high net worth customers who have a total balance of $30,000 or more from all of their accounts, list Customer ID, First and Last Name, order by descending last name

```
Select customer.ID, customer.firstName, customer.lastName
From customer
Join has on customer.ID = has.ID
Join account on has.accNumber = account.accNumber
Where account.balance >30000
Group By customer.ID
```

4. Use set operators to find non active customers, i.e. those customers who have performed no more than one transaction, list customer IDs

(unable to test as MySQL server does not support Minus and I have very unstable internet and can not maintain a stable connection to oracle)

```
Select customer.ID
From customer
Join has On customer.id = has.ID
Join performs On has.accNumber = performs.accNumber
Minus
Select performs.ID
From performs
Having Count (performs.ID) >1
```

5. Find all customers whose accounts are all held in one branch, list customer id and email address of these customers

```
Select customer.ID, customer.email
From customer
Join has On has.ID = customer.ID
Join registered On registered.accNumber = has.accNumber
Minus
(Select BSB From registered Group By BSB Having Count(*)>2)
```