Q1 – Database Design

You are working for a new music player APP called “Classic Play”.

The main function of the app is to allow customers to search the catalogue for their favourite composers, performers, pieces, or performances. For example, the composer “Ludwig van Beethoven” composed many pieces, one of which is called “Moonlight Sonata”. This piece has been recorded multiple times by different performers.

A customer can choose which performance of the piece they wish to listen to, for example: “Moonlight Sonata, played by the pianist Glen Gould, recorded live on 1st January 1960”. We want customers to be able to search by composer, piece, performer and/or date. A piece may have multiple composers, and a performance may have multiple performers. We keep composers and performers in one list of artists, since some artists may be both composers and performers, and may even perform both roles on the same performance.

To make our catalogue more visually appealing, we store and display a photo for each performance, artist. All photos are stored in the database. The sound recording itself, in MP3 format, is also stored in the database.

When someone registers to become a customer, we record their username and credit card number, and allow them to enter a short bio (maximum 500 characters), and a profile photo.

Customers can click “Like” on a performance. When others view the performance in the catalogue, they will see the number of likes. If the same customer views that performance again, they can’t “like” it more than once, but rather are offered an “unlike” button in case they change their mind.

For the above scenario, draw a ***logical*** data model. (There is no need to determine the data types of columns.)
Use crows-foot notation for relationships, and join the lines to the related columns. There is no need to add names to relationships.
You do not need to write a data dictionary.
If you wish, you may explain the reasoning behind any design decisions or assumptions you made.
Note that marks are largely based on a workable model that caters to all the constraints stated in the case. Marks may be lost for incorrect entities, relationships, cardinality or notation, lack of detail or internal contradictions.

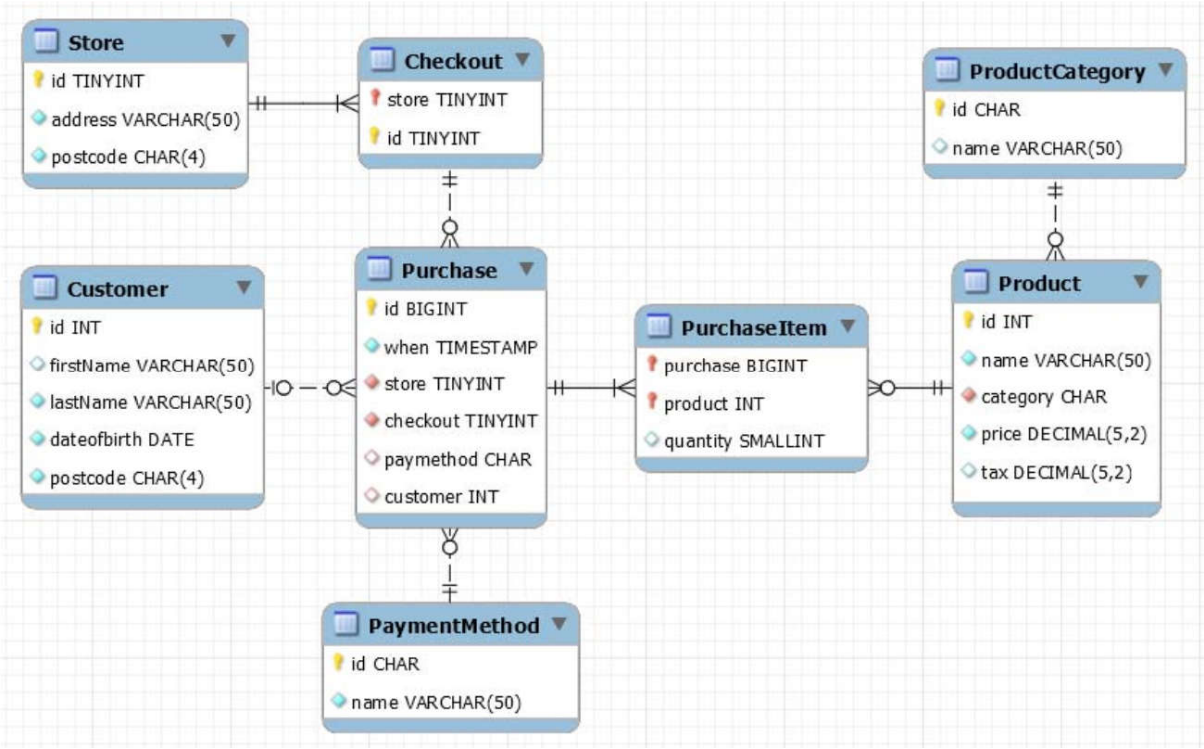
Q2 – SQL

Consider the following data model and sample data (not all data is shown) for a checkout system. Your job is to write SQL queries that answer questions posed by management.

Our customers take a collection of products to a checkout for scanning: this collection is one “Purchase”. Each item within the purchase is a “PurchaseItem”. Some customers identify themselves at the checkout by scanning a loyalty card.

The checkouts are numbered within each store: a given store n has checkouts n-1, n-2 etc. In a given store, checkout 1 is the ‘first’ checkout; the checkout with the highest number is the ‘last’. Each product is classified within a particular category. There are several payment methods.

The following ER diagram describes the database schema which has been implemented.



1. Which is the longest customer lastname that contains the letter 'E'?

2. List the names and prices of all products of type ‘food’ that cost more than $3.

   Order them by descending order of price.

3. What is the average number of items that purchases contain?

4. List the names of any customers who have used all payment methods.

5. How many purchases were made at the "last" checkout in a store?

Q3 – Normalisation

We are modelling a database for store data about food delivery. For each delivery, we must record its id number and the restaurant which is selected to provide the take-away food, including the restaurant’s id, name, phone number and address. Each delivery must be completed by one deliveryman. We must record the deliveryman’s id, name and the

selected payment method. In addition, we must record the food ordered, including its id, name and price.

Our modeller has arrived at the following relation. But this relation is not in third normal form.

DELIVERY (deliveryId, restaurantId, restaurantName, restaurantPhone, restaurantAddress, deliverymanId, deliverymanName, paymentMethod, (foodId, foodName, foodPrice))

Your job is to convert the relation to 3rd normal form.
Mark your primary keys with a solid underline, and your foreign keys with a dotted underline. (Any attributes that are both primary and foreign keys should get both underlines.)
You don't need to show intermediate normal forms – just the 3rd normal form you end up with.

Q4 – Physical Design

For each of the following columns in the database shown in question 1, write the column's MySQL data type, include the width of the column where appropriate.

1. Performance.id
2. Performance.when
3. Performance.photo
4. Performance.recording
5. Artist.name
6. Like.when
7. Customer.username
8. Customer.creditcard
9. Customer.bio

Q5 – Transactions

A) Explain in which situation can create **inconsistent retrieval** problem and give an example.
B) List five different lock granularities, explain their meaning and when to use them.

Q6 – Applications

A) List three different logics in Database System Architecture and its main components.
B) Explain the advantages and disadvantages of 3-tier architecture.

Q7 – Database Administration

A) List threats to databases and provide some solutions.

B) Compare the difference between hot backup and cold backup.

C) How can we protect against “SQL Injection Prevention”?

Q8 – Distributed Databases

A) Compare distributed database with decentralized database.
B) List four advantages and disadvantages of distributed database.
C) List three disadvantages of replication.

Q9 – NoSQL

A) Give an example of NoSQL and explain its advantage over relational database.
B) Explain what is CAP theorem in detail.