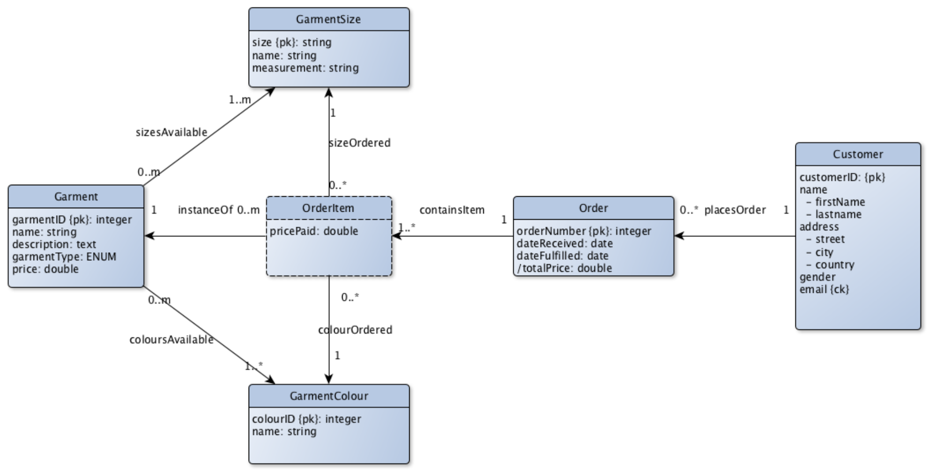
F21DF Coursework 1 Feedback

The work was generally well done with over half of the course getting 70% or above.

Below you will find my sample answers together with generic feedback about the submissions received.

Conceptual Model

My sample answer:



Alternative answers are possible.

Feedback on your answers:

Answers generally captured the main entities to model the basic data requirements but several missed the more complex requirements, e.g. relationships to the OrderItem from colour and size. Many diagrams did not include type, size, and colour information in purchased items. Direct relationships between these entities and the order item entity are required.

Many models also included attributes that belong to other entities, such attributes should be modelled with a relationship in an ER diagram. **Foreign keys should never appear on an ER diagram;** they are a feature of the relational data model.

Every attribute should have a data type indicated.

Every relationship should have a label associated with it. You should try to have unique relationship names so that they can be used if necessary for table names.

Total price of an order should be a derived attribute from the total of the price paid for each item.

Relational Model

For this part I was looking to check that you could follow the 7 steps for converting your ER diagram into a data dictionary and that you selected suitable data types. If you needed to deviate from the result of the 7 steps, then this deviation should be explained.

Feedback on your answers:

Descriptions should be more than just a repetition of the column name. Their purpose is to document the attribute.

Proper consideration of datatypes for columns, e.g. `VARCHAR(255)` is not appropriate for `gender` , an `ENUM` should be used. Only use the `TEXT` type if you need more than 255 characters; the implementation of `TEXT` is not as efficient as `VARCHAR` and `CHAR`. Care should also be given to the use of `DATETIME` and `DATE`, for this database you only needed the date.

Foreign key reference column should state the table and attribute that is referenced.

Relational Schema

For this part I was ensuring that you could create tables in MySQL and populate them with data. I was looking for primary and foreign key constraints to be defined.

Feedback:

Several of your designs, particularly for Garments with relation to Size and Colour, resulted in the duplication of data related to the Garment when the Size or Colour changed. This is an indication of poor database design and should lead you to rethink the database design. **Duplication of data is highly problematic since it is likely to deviate between entries.**

Foreign key reference action on delete and update should be considered and made explicit. Foreign keys should be declared before inserting data. For example, what should happen to a fulfilled order item if the customer is deleted?

The total amount of an order can be calculated on insert using an update query.

Multiple rows of data can be included in a single insert statement.

While large amounts of data were not required, the data inserted should demonstrate the multiplicity of relationships, and include the features that were appearing in the queries.

Queries

For this part I was checking that you could write SQL queries to answer specific questions. You should ensure that you only return the columns asked for.

Below are the queries that I would expect based on my ER diagram. Some of your answers deviate from this due to your model. No marks were deducted if your query was correct for your model.

Queries a and d were generally correct across the board. Query b was most of the time correct. Queries c and e were more problematic.

a) Name and email of customers who have unfulfilled orders

```
SELECT C.firstname, C.lastname, C.email
FROM   Customer AS C, Orders AS O
WHERE  O.customerID = C.customerID AND
       dateFulfilled IS NULL
```

Feedback

When returning names of people you should always give first and last name unless otherwise specified. Some students chose to concatenate these columns which is fine.

If you included a fulfilled flag in your design then you would test for that flag rather than a NULL status. However, this is duplicating information which is something that should be avoided in a relational database.

b) Garments available in 5 or more colours

```sql
SELECT G.name, COUNT(*) as colourCount
FROM Garment AS G, ColoursAvailable AS C
WHERE G.garmentID = C.garmentID
GROUP BY C.garmentID
HAVING COUNT(*) >= 5
```

Or alternatively using a sub-query

```sql
SELECT G.name, C.colourCount
FROM   Garments AS G,
      (SELECT garmentID, COUNT(*) AS colourCount
       FROM   ColoursAvailable
       GROUP BY garmentID
       HAVING COUNT(*) >= 5
      ) AS C
WHERE  G.garmentID = C.garmentID
```

Feedback

Several of you missed the `HAVING` clause to restrict the count.

c) Orders that do not contain trousers

This query requires a not exists clause. Look at the lecture queries about finding projects that Gordon Smith has not worked on.

```sql
SELECT O.orderNumber
FROM   Orders AS O
WHERE  O.orderNumber NOT IN
        (SELECT I.orderNumber
         FROM   OrderItem AS I, Garment AS G
         WHERE  G.garmentID = I.garmentID AND
                G.garmentID = 'TROUSER')
```

Feedback

You should not assume knowledge of the identifier for trousers. You need to have a string match, but it can be an equality match. You could equally use the `LIKE` keyword.

d) Total income from orders on specific date

```sql
SELECT SUM(pricePaid) AS TotalIncome
FROM   Orders AS O, OrderItem AS I
WHERE  dateReceived = '2020-09-01' AND
       O.orderNumber = I.orderNumber
```

If you wanted to do this for the current date instead, the query would be

```sql
SELECT SUM(pricePaid) AS TotalIncome
FROM   Orders AS O, OrderItem AS I
WHERE  dateReceived = CURDATE() AND
       O.orderNumber = I.orderNumber
```

Feedback

When looking for a specific date you should use equality, and if need be date functions, rather than the `LIKE` function which requires string comparisons which are inefficient.

e) Customer with longest wait

There could be more than one customer with the most expensive purchase item. The query requires the `MAX` function. I've shown it as a nested query here but if cost is pre-computed in your database then you can do it as a simple query joining guest and booking as in the second query below.

```sql
SELECT C.firstname, C.lastname
FROM   Customer AS C,
       (SELECT customerID, MAX(dateFulfilled -
dateReceived)
        FROM   Orders) as M
WHERE  c.customerID = M.customerID
```

Feedback

This was the poorest performed query. Several students returned the most recently placed orders, while others used `ORDER BY` with a `LIMIT` of 1. However, this does not permit multiple customers to have the longest wait.