BREAKDOWN OF TASK:

* **Paper based order record system to Local Database System**

Build pipeline to extract, transform, load data (OR ELT)

Extract – Get Info from the csv

Transform – Clean Data

Load – Into a Database (MYSQL)

* **CLI Application that allows company to trigger ETL process**

Accept User input

Clear screen

Print to screen

Trigger all 3 stept of ETL

* **GUI Application that allows company to trigger ETL process**
* **MUST CREATE OWN DATA, WITH STRUCTURE:**
  + **Products** - Customer Name, Drink, Qty, Price, Branch, Payment Type, Card Number, Date/Time
  + *Dave, “Latte, 2, £3.50”, Epsom, Card, 0123456, 12/08/2024*

**Tools:**

* Python
* Docker
* MYSQL database

**Final Deliverables to the Client**

1. Deliver a Product demo of the applications functions (5 Mins)
2. Create and Deliver a Client Facing Presentation on the benefits to the customer of you application (5 Mins)
3. Create and Deliver a White board session with the Client, explaining how the Application Works and your design decisions with alternatives. (5 mins

**Source Control** - You must use GIT and GIT hub

MY PLAN:

Schema For data

* **Name (ex. Ruth Fashogbon). – will not have data in database, will be represented as id**
* First Name and Last Name –Fill as None
* Not blank – Fill as None
* Check Whitespaces
* **Drink Ordered (ex. Espresso)**
* Check Whitespace
* Correct Spelling – get list of all drinks in list and other ckey café drinks to check spelling against
* Check blank - if blank remove??
* Split Price out of it
* Make sure price is all in same data type (float)and formay (NO £)
* **Qty (ex. 2)**
* Check Integer not String or Float
* If Blank and price = 1 of that item auto fill as “1”.
* Check Whitespace
* **Branch (ex. Epsom)**
* Check Spelling
* Check Whitespace
* Check blank - if blank remove??
* **Payment Type(ex. Card)**
* Only 2 option Card/Cash
* Check Spelling
* Check whitespace
* Check blank - if blank remove??, or check if card number if so = card, if not = cash
* **Card Number (ex. 1234567890987654) – can get rid of info (too sensitive)**
* Integer only
* Max length 16
* Min 15
* Check whitespace
* Check all card type has card number
* **Date/Time (14/07/2025 12:48)**
* Check whitespace
* Correct date format dd/mm/yyyy
* Product
* Primary Key – Product ID use an auto-increment or UUID
* Product (drink\_name)
* Price
* Order
* Primary Key – Order ID use an auto-increment or UUID
* Product ID
* Qty
* Price
* Total Amount
* Transaction
* Primary Key - Transaction ID use an auto-increment or UUID

**Product Table**

* Contains details about each individual product (drink) such as:
  + **Drink ID** (unique identifier for the product)
  + **Drink Name**
  + **Price**
* This table is *independent* of any customer or order information. It purely stores product info.

**Order Table**

* Represents a **customer's entire order**.
* Contains details that apply to the whole order, such as:
  + **Order ID** (unique identifier for the order)
  + **Date/Time** of the order
  + **Branch** where the order was placed
  + **Payment Type** used for the order
* This table doesn’t detail what was ordered—just the order itself and related metadata.

**Order Item Table**

* Breaks down the **individual products within an order**.
* Each row represents a line item on an order:
  + **Order Item ID** (unique line item ID)
  + **Order ID** (links back to the main order)
  + **Drink ID** (which product was ordered)
  + **Quantity** (how many of that product were ordered)
* This allows you to see *exactly* what and how much was ordered per order.