Part 1

Assuming a simple DB structure of 5 tables:

Product with columns `(product_id, model, upc, quantity, manufacturer_id)`

Manufacturer with columns `(manufacturer_id, name)`

Customers with columns `(customer_id, first_name, last_name, email_address)`

Orders with columns `(order_id, customer_id)`

`Order_products` with columns `(order_id, product_id, qty)`

Please provide the queries you would have used for the following (each of them should be done in a single query):

1. **Give me a list of products that belongs to manufacturer Blanco.**

```
-- Query: List of products belonging to manufacturer Blanco

-- Description: The "Blanco" manufacturer's items are all retrieved by this query.


SELECT p.*

FROM Product p

JOIN Manufacturer m ON p.manufacturer_id = m.manufacturer_id

WHERE m.name = 'Blanco';
```

2. **Get me only Blanco `model` and `upc` columns from `product` table that has positive quantity.**

```
-- Query: Blanco model and UPC with positive quantity

-- Description: The'model' and 'upc' columns from the 'product'
database are retrieved for goods with a positive quantity that belong
to the 'Blanco' manufacturer.


SELECT p.model, p.upc

FROM Product p

JOIN Manufacturer m ON p.manufacturer_id = m.manufacturer_id

WHERE m.name = 'Blanco' AND p.quantity > 0;
```

3. **Get me manufacturer, model, and quantity bought by customer with an email of blipps@trumbull.com**

```
-- Query: Manufacturer, model, and quantity bought by customer with
email blipps@trumbull.com

-- Description: This query returns information on the brand, model,
and number of goods purchased by a user with the email address
blipps@trumbull.com.


SELECT m.name AS manufacturer, p.model, op.qty AS quantity

FROM Product p

JOIN Manufacturer m ON p.manufacturer_id = m.manufacturer_id

JOIN Order_products op ON p.product_id = op.product_id

JOIN Orders o ON op.order_id = o.order_id

JOIN Customers c ON o.customer_id = c.customer_id

WHERE c.email_address = 'blipps@trumbull.com';
```

Part 2

Bonus: Provide a logic flow of the needed Procedure that will be created to solve this problem:

Challenge Question - We want to generate an inventory age report which would show the distribution of remaining inventory across the length of time the inventory has been sitting at the warehouse. We are trying to classify the inventory on hand across the below 4 buckets to denote the time the inventory has been lying in the warehouse.

- 0-90 days old
- 91-180 days old
- 181-270 days old
- 271 – 365 days old

For example, the warehouse received 100 units yesterday and shipped 30 units today, then there are 70 units which are a day old.

The warehouses use FIFO (first in first out) approach to manage inventory, i.e., the inventory that comes first will be sent out first.

Table 1

ID	OnHandQuantity	OnHandQuantityDelta	event_type	event_datetime
TR0013	278	99	OutBound	25/05/2020 00:25
TR0012	377	31	InBound	24/05/2020 22:00
TR00ll	346	1	OutBound	24/05/2020 15:01
TR00l0	346	1	OutBound	23/05/2020 05:00
TR009	348	102	InBound	25/04/2020 18:00
TR008	246	43	InBound	25/04/2020 02:00
TR007	203	2	OutBound	25/02/2020 09:00
TR006	205	129	OutBound	18/02/2020 07:00
TR005	334	1	OutBound	18/02/2020 08:00
TR004	335	27	OutBound	29/01/2020 05:00
TR003	362	120	InBound	31/12/2019 02:00
TR002	242	8	OutBound	22/05/2019 00:50
TR00l	250	250	InBound	20/05/2019 00:45

For example, on 20th May 2019, 250 units were inbounded into the FC. On 22nd May 2019, 8 units were shipped out (outbound) from the FC, reducing inventory on hand to 242 units. On 31st December, 120 units were further inbounded into the FC increasing the inventory on hand from 242 to 362.On 29th January 2020, 27 units were shipped out reducing the inventory on hand to 335 units.

On 29th January, of the 335 units on hands, 120 units were 0-90 days old (29 days old) and 215 units were 181-270 days old (254 days old).

Columns:

ID of the log entry

`OnHandQuantity`: Quantity in warehouse after an event

`OnHandQuantityDelta`: Change in on-hand quantity due to an event

`event_type`: Inbound – inventory being brought into the warehouse; Outbound – inventory being sent out of warehouse.

`event_datetime`: date- time of event

The data is sorted with latest entry at top.

Sample output:

0-90 days old	91-180 days old	181-270 days old	270-365 days old
176	102	0	0

Table 2

Solution:

1. Let's begin by initializing four variables that we will be using for our calculations. These variables are `count_0_90_days, count_91_180_days, count_181_270_days and count_271_365_days`. It is important to set all these variables to zero in order to avoid any potential errors in our future calculations.

```
# Initialize count variables

count_0_90_days = 0

count_91_180_days = 0

count_181_270_days = 0

count_271_365_days = 0
```

2. Iterate through each entry in Table 1, starting from the latest entry:
    a. Check the `event_type` of the entry.
    b. If the `event_type` is "Inbound," add the `OnHandQuantityDelta` to the respective count based on the age range determined by the `event_datetime`.
    c. If the `event_type` is "Outbound," subtract the `OnHandQuantityDelta` from the respective count based on the age range determined by the `event_datetime`.

(Throughout the solution I will be referring the inventory table as Table 1 and the sample output table as table 2)

```
for entry in reversed(table1):
    event_type = entry['event_type']
    event_datetime = entry['event_datetime']
    on_hand_quantity_delta = entry['OnHandQuantityDelta']

    # Based on the event_datetime, determine the age range
and calculate the days_since_event

    if event_type == 'Inbound':
        if days_since_event <= 90:
            count_0_90_days += on_hand_quantity_delta
        elif days_since_event <= 180:
            count_91_180_days += on_hand_quantity_delta
        elif days_since_event <= 270:
            count_181_270_days += on_hand_quantity_delta
        elif days_since_event <= 365:
            count_271_365_days += on_hand_quantity_delta
    elif event_type == 'Outbound':
        if days_since_event <= 90:
            count_0_90_days -= on_hand_quantity_delta
        elif days_since_event <= 180:
            count_91_180_days -= on_hand_quantity_delta
        elif days_since_event <= 270:
            count_181_270_days -= on_hand_quantity_delta
        elif days_since_event <= 365:
            count_271_365_days -= on_hand_quantity_delta
```

3. The counts in the four variables will show how the leftover stock is distributed across the designated age buckets after processing all of Table 1's entries.

4. Create Table 2 with the following age ranges as columns: 0-90 days, 91-180 days, 181-270 days, and 271-365 days.

```
table2 = {
    '0-90 days old': count_0_90_days,
    '91-180 days old': count_91_180_days,
    '181-270 days old': count_181_270_days,
    '271-365 days old': count_271_365_days
}
```

5. Add the counts from Step 1 to Table 2. Each count should be put in the appropriate age range column.

By combining the information from Table 1 and the logic flow described above, we can create Table 2 and get an inventory age report that displays how the remaining inventory is distributed across the designated age buckets.

Sample Output (Table 2):

Age Range	Count
0-90 days old	120
91-180 days old	0
181-270 days old	215
271-365 days old	0

This table shows that there are 215 units in the 181–270-day age range and 120 units in the 0–90-day age range of inventory.

Implementation→ GitHub