Candidate Number _____ (Tej B.) – I don't know my # Sorry

ER Conceptual Model) In my ER conceptual models I created the entities Customer, Payment, Basket, Product, Orders, Review, Delivery, and Returns to address all aspects of the description of the E-commerce database. For these entities, I had multiple attributes for to represent all the information that was necessary to be stored in the entities in order to create the desired database with enough information to answer questions 2-6. (i) For my conceptual model, I only had 1 use of specialization, and it was within my payment entity. This was important because I needed to create subtypes of Voucher/Gift and Debit/Credit so I could provide them with the necessary personalized supertype attributes to allow the database to pick up different reads/ways of analyzing the two different payment methods. This represented a total specialization because all supertype attributes connected to either the Voucher/Gift or Debit/Credit subtype. (ii) For each of my relationships I had the following cardinalities:

- Customer Pays Payment: optional and many from customer to payment because it is possible for a customer not to make a purchase (making the payment optional) and because they could make multiple different types of payments. For payment to customer I had mandatory because a made payment needs an associated customer and 1 because each payment type can only be associated with 1 customer at a time.
- Customer Provides Review: optional and many from customer to review because a customer is not required to make a review and because they can make as many reviews as they desire. I had mandatory and one from review to customer because each review must have an associated customer who made it and because a review with its individual review_# can only be made by 1 customer.
- Customer Places Orders: I made it optional and many from customer to order because it's not required for a customer to make an order (can just be making a basket) and because they can also submit multiple orders. For orders to customer I made it mandatory and one because each order with its personalized order_# can only be made by one customer and because an order must've been placed by a customer which makes it mandatory.
- Customer Creates Basket: I made it optional and many from customer to basket because a customer is not required to create a basket and because a customer can also make multiple different baskets. For basket to customer, I made it mandatory and one because a basket needs to have been made by a customer and because a basket with its individual basket_ID can only have one associated customer.
- Basket Contains Product: For basket to product, I made it mandatory and many because a basket needs to have at least 1 product in its wish list but it can also have as many as the person wants in there. For product to basket I made it optional and many because a product doesn't necessarily have to be in a basket but the same product can also be in multiple different baskets.
- Product Got_reviewed Review: For product to review, I made it optional and many because a product doesn't necessarily have to have a review and also because a product can have multiple reviews. For review to product, I make it mandatory and one because a review must have an associated product its talking about and because it can only be talking about 1 product in the review.

- Product Has Orders: For product to orders, I made it optional and many because a product doesn't necessarily need to be part of an order (can be in a basket or nothing at all) and because a product can also be part of multiple orders. For orders to product, I made it mandatory and many because an order needs to have at least 1 product in it and because an order can also have multiple of the same or different products within it.
- Orders Returned Returns: For orders to returns, I made it optional and many because an order doesn't necessarily have to be returned and because my assumption was that an order can be attempted be returned twice if it was denied the first time. For returns to orders, I made it mandatory and one because a return has to have an associated order and because a return with its individualized ticket number can only be returned back from 1 order.
- Orders Deliver Delivery: From orders to delivery, I made it mandatory and one because an order needs to be delivered to its associated customer and because an individual order can only be delivered once successfully. For delivery to order, I made it mandatory and one because a delivery needs to be containing an associated order and a delivery with its personalized tracking_# can only contain 1 order even it has multiple products within the number.

(iii) In my conceptual model, there are some attributes within entities that are not mandatory to the model. For example, the attribute Phone_# is optional and may not be necessary to whether a customer is able to make the purchase and get the delivery or not. Or maybe in review, maybe the customer doesn't provide a review_text, this wouldn't really be mandatory. However, there are some entities in my model where all attributes would be mandatory. For example, in Payment, for either payment method, it is required for all information (attributes) to be filled out.

(iv) Some other things that should be mentioned is my use of a composite attribute for address which allowed me to add other attributes important to the address information and my uses of unique variables for all individual number types (order#, ticket#, product#, etc) because they are 1 of 1.

Relational Model) (i) There were a few changes I had to make to my relational model in order to make the database work out in the end. I had to switch the arrows from Payments → voucher/gift and debit/credit to Payments ← voucher/gift and debit/credit in order to create foreign keys of Payment method so the EDR would generate the right SQL necessary for linking the payment method to payment. In payment I also had to primary key payment_method and customer_email in the Payments Table. Along with this I changed the Contains Table to Basket_contains just to add more clarity to my model. I also flipped the arrow so that basket_contains was pointing to both Basket and Product so that Product_number and Basket_ID were both foreign keyed so that they would be able to connect in the database I later generated. (ii) I don't believe I had any views. (iii) The constraints related to cardinalities were all the same as before in the ER conceptual model. (iv) I added quantity, subtotal, and fee to the HAS table in order to generate better SQL that would be better at answering a few question in 5.1-5.4. All other changes were specified in my answer in (i).

Database Creation) My database creation was very quick and straightforward. Once I entered the copy/pasted data in from my EDR relational model there were really no errors in which I had to navigate or make any changes for. My order of tables was CUSTOMER, PRODUCT, BASKET, PAYMENT, DEBIT_O_CREDIT, VOUCHER_O_GIFT, REVIEW, BASKET_CONTAINS, DELIVERY, ORDERS, HAS, RETURNS.

Data Creation) For my data creation I used synthetic data to populate my database. To gather this data, I used the AI tool, ChatGPT, to help create random data for my tables. For each table I had, I gave the AI tool all the different attributes of the table and had it fill in the data for 15 different customers and products. I then had it randomly generate the other attribute values of the different tables based of the information it had that needed to be the same and the new information it had to add to populate the data sets. Although this definitely helped me, I was still getting multiple errors when I run the directly given code. To ensure the data was consistent to my model I went through all the different foreign key data that needed to be the same in different tables and changed some of the errors so that all the foreign keys were exactly accurate and in the same order as the other ones in different tables. Finally, after lots of effort and trial, the database was able to populate successfully.

5.1)

(i) Question 5.1

(ii) For question 5.1 I was asked to generate a query that shows the customer's name and email, order number, order date, the list of products in each order and the total of each order. To do this I used the select function to select the needed information, and then I used to Group_concat function to allow the program to aggregate the product information. I used the from and join clauses to connect the necessary tables. Finally I used the the group by and order by clauses to group all the customers together and sort them by name and order date.

(iii)

	Customer_Name	Customer_email	Order_number	Order_date	Products_Ordered	Total_Paid
1	Ahmed Hassan	ahmed.hassan@email.com	10	2024-07-18	Radiance Serum (1)	34.99
2	Alice Johnson	alice.johnson@email.com	3	2024-03-13	Mystery Book (1)	24.99
3	Bob Brown	bob.brown@email.com	4	2024-04-08	Power Blender (1)	69.99
4	Carlos Silva	carlos.silva@email.com	14	2024-09-16	Barista Pro (1)	249.99
5	David Kim	david.kim@email.com	8	2024-06-18	Smart Air Fryer (1)	119.99
6	Elena Petrova	elena.petrova@email.com	13	2024-09-01	Rustic Desk (1)	389.99
7	Emma Wilson	emma.wilson@email.com	5	2024-05-03	Pro Racket (1)	149.99
8	Jane Smith	jane.smith@email.com	2	2024-02-23	Classic Jeans (2)	109.98
9	Jane Smith	jane.smith@email.com	17	2024-10-30	Classic Jeans (2)	109.98
10	John Doe	john.doe@email.com	1	2024-01-18	Pro Laptop (1)	999.99
11	John Doe	john.doe@email.com	16	2024-10-15	Pro Laptop (1), Premium Headphones …	1289.98
12	Liam Johnson	liam.johnson@email.com	12	2024-08-17	Comfort Pet Bed (1)	89.99
13	Maria Garcia	maria.garcia@email.com	9	2024-07-03	Pro Dumbbells (1)	199.99
14	Michael Chen	michael.chen@email.com	6	2024-05-18	Premium Headphones (1)	289.99
15	Olivia Taylor	olivia.taylor@email.com	11	2024-08-02	NextGen Console (1)	489.99
16	Sophie Martin	sophie.martin@email.com	7	2024-06-03	Ceramic Planter (1)	34.99
17	Yuki Tanaka	yuki.tanaka@email.com	15	2024-10-01	HealthTrack Watch (1)	189.99

5.2

(i) Question 5.2

(ii) Similar to question 5.1, I used the select clause to select different columns and data from different tables. Then I used the Coalesce function to handle null values from gift card balances, I spent a lot of time looking up functions that would help me do this. Then I used the Group_concat function to put the product information together like in the last question. I also ended up using the Left Join functions because it made it possible to include gift less and basket less customers. Finally, I used a similar ending structure as my last question except I added a Having clause to eliminate customers without baskets.

(iii)

	Customer_email	Name	Birthdate	Gift_Card_Balance	Products_In_Basket
1	ahmed.hassan@email.com	Ahmed Hassan	1994-08-27	0	Radiance Serum (1)
2	alice.johnson@email.com	Alice Johnson	1992-03-08	50	Mystery Book (1)
3	bob.brown@email.com	Bob Brown	1988-12-01	75	Power Blender (1)
4	carlos.silva@email.com	Carlos Silva	1996-05-25	80	Barista Pro (2)
5	david.kim@email.com	David Kim	1991-02-14	0	Smart Air Fryer (1)
6	elena.petrova@email.com	Elena Petrova	1984-12-09	0	Rustic Desk (1)
7	emma.wilson@email.com	Emma Wilson	1995-07-30	0	Pro Racket (1)
8	jane.smith@email.com	Jane Smith	1985-10-20	0	Classic Jeans (2)
9	john.doe@email.com	John Doe	1990-05-15	0	Pro Laptop (1)
10	liam.johnson@email.com	Liam Johnson	1997-01-12	30	Comfort Pet Bed (1)
11	maria.garcia@email.com	Maria Garcia	1989-06-18	100	Pro Dumbbells (3)
12	michael.chen@email.com	Michael Chen	1987-09-22	0	Premium Headphones (2)
13	olivia.taylor@email.com	Olivia Taylor	1986-04-03	0	NextGen Console (2)
14	sophie.martin@email.com	Sophie Martin	1993-11-05	25	Ceramic Planter (1)
15	yuki.tanaka@email.com	Yuki Tanaka	1993-07-19	0	HealthTrack Watch (1)

5.3

(i) Question 5.3

(ii) First, I created the ProductSales by using the Select clause to gather the necessary data/columns from the various tables, and the Sum function to make the TotalQuantitySold and TotalSales columns that added the subtotals and quantities together from the HAS table. Then I used the partition to separate the categories by sums of both TotalQuanititySold and TotalSales. From there I used the RankInCategory as <= 2 in order to get only the top two products in each category. However, because for some of my input data their were multiple ties in the amount of products sold, for some of the categories they're 3 products which are all tied in quantities sold. I didn't really know how to fix/change this with out altering a lot of my foreign keys which kept creating errors in my SQL.

(iii)

	Category	Product_name	TotalQuantitySold	TotalSales
1	Beauty	Radiance Serum	1	39.99
2	Books	Mystery Book	1	24.99
3	Clothing	Classic Jeans	4	239.96
4	Electronics	Pro Laptop	2	1999.98
5	Electronics	Premium Headphones	2	599.98
6	Furniture	Rustic Desk	1	399.99
7	Home & Garden	Ceramic Planter	1	34.99
8	Home & Kitchen	Smart Air Fryer	1	129.99
9	Home & Kitchen	Power Blender	1	79.99
10	Pets	Comfort Pet Bed	1	89.99
11	Sports	Pro Racket	1	149.99
12	Sports	Pro Dumbbells	1	199.99

5.4

(i) Question 5.4

(ii) First I created monthly_sales in order to calculate the total_sales in a given month. ‘strftime()’ was used in the google search I saw as a tool to extract the month and year from order dates. Using this along with the from, join, sum clauses that I used in pervious questions I was able to add up the orders by month and sort them by month/year. I also had to make sure that returned items didn’t affect the month’s sales I set subtotal to 0 if the status from return was completed. Finally I took this information and created the formula get the total sales growth. I was able to find a structured formula for growth on the internet but I still had to rearrange it and plug in the variables I created like total-sales.

(iii)

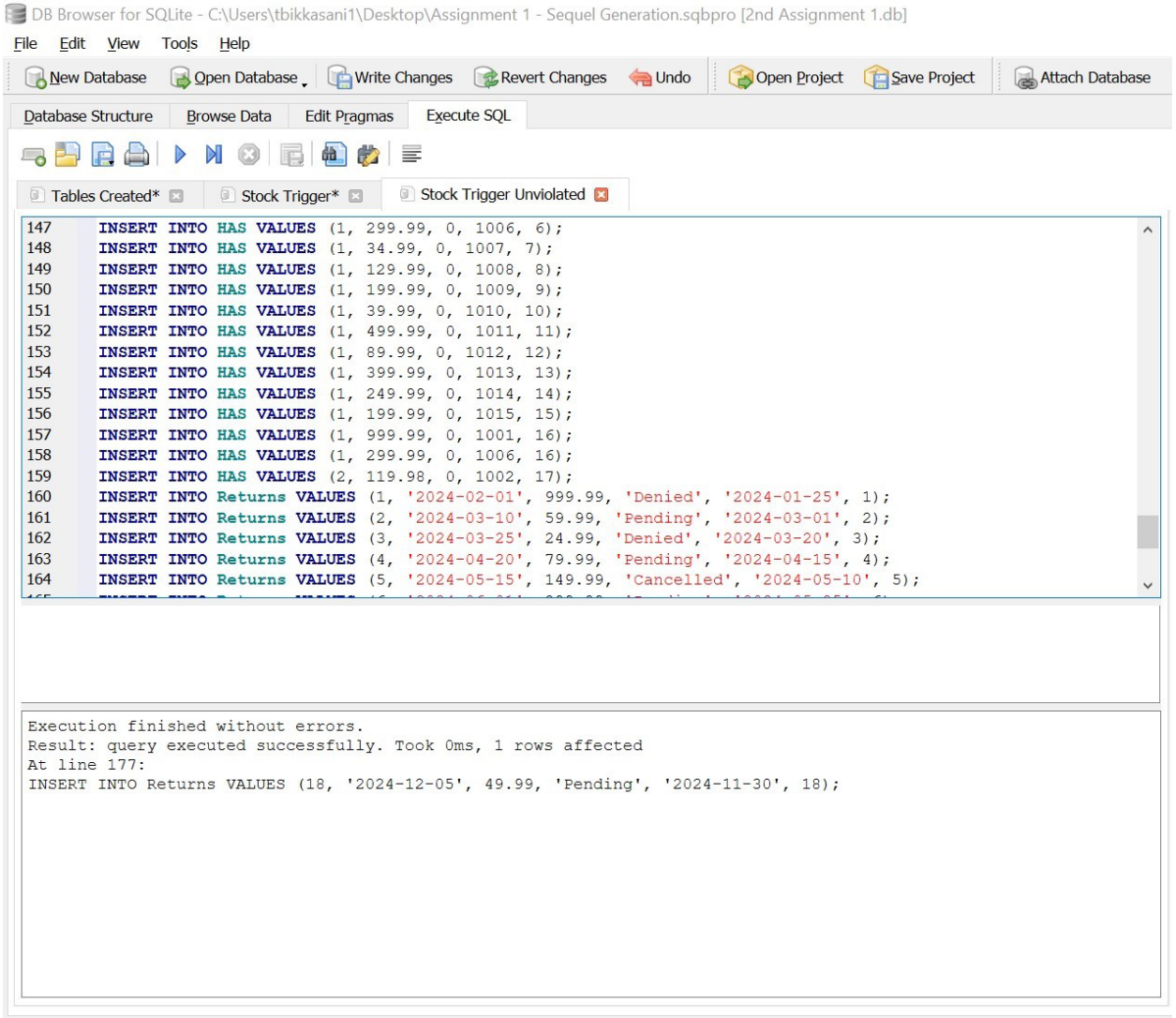
	year	month	total_sales	sales_growth_percentage
1	2024	01	999.99	*NULL*
2	2024	02	119.98	-88.0018800188002
3	2024	03	24.99	-79.171528588098
4	2024	04	79.99	220.088035214086
5	2024	05	449.98	462.545318164771
6	2024	06	129.99	-71.1120494244189
7	2024	07	199.99	53.850296176629
8	2024	08	589.98	195.004750237512
9	2024	09	249.99	-57.6273771992271
10	2024	10	1619.95	548.005920236809

Question 6

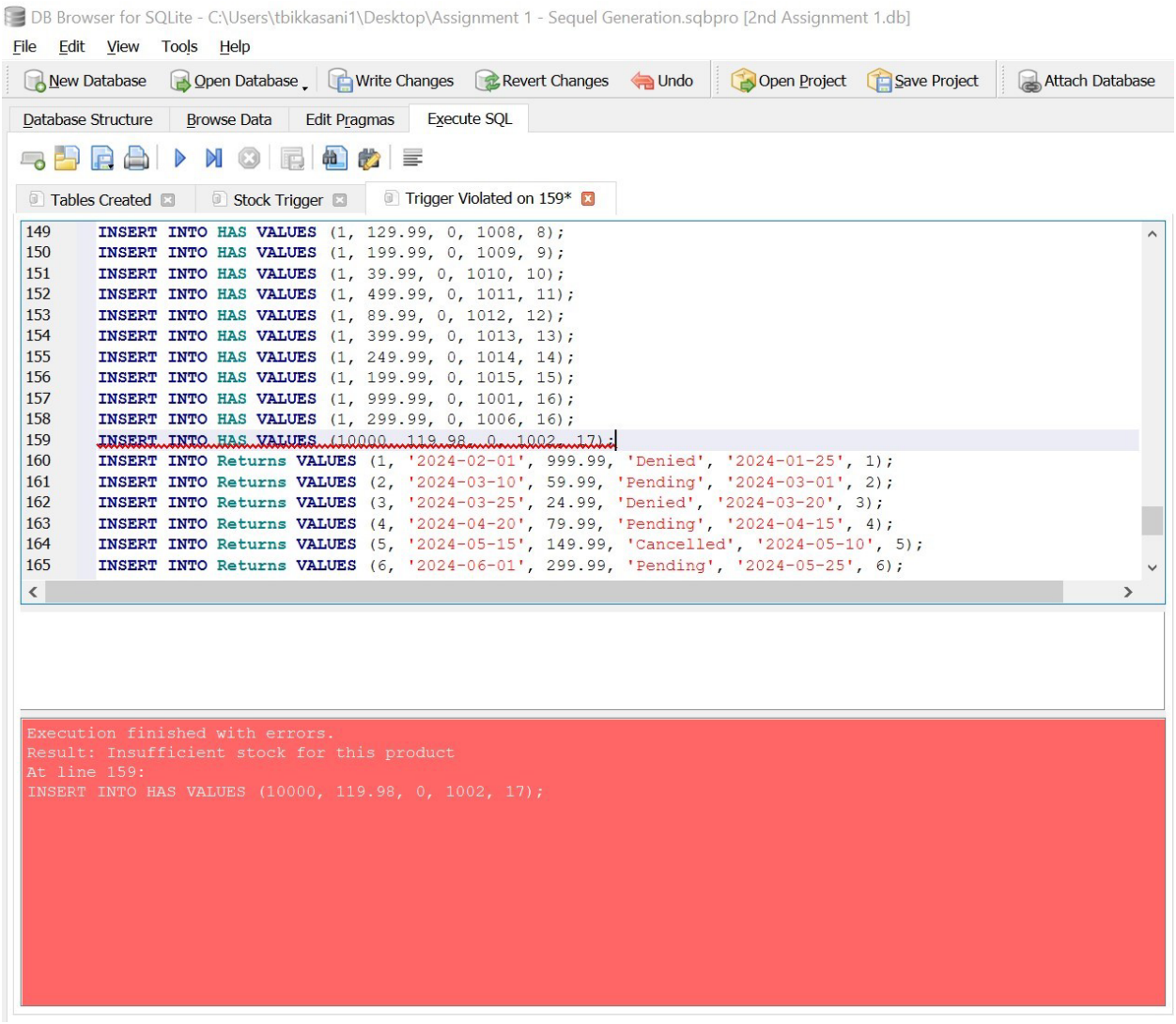
(i) Checking whether a product is in stock when the consumer places an order

(ii) For question 6, I first started by creating and naming trigger using Check Trigger to create the check_stock trigger. The stock attribute was within the HAS table so I created a “Before Insert on Has” to select a case for when the stock attribute would be triggered. Basically just said that when the quantity selected for a product is higher than the product number then error should be “insufficient stock for this product”. Lastly I just put END to end the trigger. To test this trigger I input a HAS row with a stock of 2 and then a Has row with a stock of 10000. It gave the error on the latter option which meant the trigger was working correctly.

(iii) Trigger Unviolated



(iv) Trigger Violated


DB Browser for SQLite - C:\Users\tbikkasani1\Desktop\Assignment 1 - Sequel Generation.sqbpro [2nd Assignment 1.db]
File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Database
Database Structure Browse Data Edit Pragmas Execute SQL
Tables Created Stock Trigger Trigger Violated on 159*
149 INSERT INTO HAS VALUES (1, 129.99, 0, 1008, 8);
150 INSERT INTO HAS VALUES (1, 199.99, 0, 1009, 9);
151 INSERT INTO HAS VALUES (1, 39.99, 0, 1010, 10);
152 INSERT INTO HAS VALUES (1, 499.99, 0, 1011, 11);
153 INSERT INTO HAS VALUES (1, 89.99, 0, 1012, 12);
154 INSERT INTO HAS VALUES (1, 399.99, 0, 1013, 13);
155 INSERT INTO HAS VALUES (1, 249.99, 0, 1014, 14);
156 INSERT INTO HAS VALUES (1, 199.99, 0, 1015, 15);
157 INSERT INTO HAS VALUES (1, 999.99, 0, 1001, 16);
158 INSERT INTO HAS VALUES (1, 299.99, 0, 1006, 16);
159 INSERT INTO HAS VALUES (10000, 119.98, 0, 1002, 17);
160 INSERT INTO Returns VALUES (1, '2024-02-01', 999.99, 'Denied', '2024-01-25', 1);
161 INSERT INTO Returns VALUES (2, '2024-03-10', 59.99, 'Pending', '2024-03-01', 2);
162 INSERT INTO Returns VALUES (3, '2024-03-25', 24.99, 'Denied', '2024-03-20', 3);
163 INSERT INTO Returns VALUES (4, '2024-04-20', 79.99, 'Pending', '2024-04-15', 4);
164 INSERT INTO Returns VALUES (5, '2024-05-15', 149.99, 'Cancelled', '2024-05-10', 5);
165 INSERT INTO Returns VALUES (6, '2024-06-01', 299.99, 'Pending', '2024-05-25', 6);
Execution finished with errors.
Result: Insufficient stock for this product
At line 159:
INSERT INTO HAS VALUES (10000, 119.98, 0, 1002, 17);