Official (Open)

# Declaration of Academic Integrity

Academic Integrity is a central tenet of Singapore Polytechnic. The polytechnic rules state that "Cheating in examinations and other assessed work is a very serious offence. This includes copying and using plagiarised material. Any student who cheats, attempts to cheat or breaches any rules for examinations and tests will face disciplinary action. The student is liable to be expelled."

Check **only one** of the two options below:

I affirm that the work I submit is my own, produced without help from any AI tool(s) and/or other source(s).

I affirm that the work I submit has been produced with the use of AI tool(s) and/or other source(s) which I have acknowledged fully in the following section.

By signing this form, I declare that the above affirmation made is true, and that I have read and

understood the rules stated in Students Handbook on “[Plagiarism](https://www.sp.edu.sg/sp/student-services/osc-overview/student-handbook/intellectual-property-copyright-and-plagiarism)” and “[Breach of Examination/Assessment Rules](https://www.sp.edu.sg/sp/student-services/osc-overview/student-handbook/conduct-in-examinations-breach-of-exam-rules)”.

|  |  |  |
| --- | --- | --- |
| Name:  Lois Poh | Student ID:  2429478 | Class:  DIT2B21 |
| Signature:  X | Date: 15/8/2025 |  |

Singapore Polytechnic School of Computing Page **1** of **2**

Official (Open)

Acknowledgement of use of AI tool(s) and/or other source(s) (where applicable)

AI tool(s)

|  |  |
| --- | --- |
| **Name of AI tool** | ChatGPT |
| **Input prompt** | can you give me queries for when the discount is percentage or fixed or like when the discount isn't active |
| **Date generated** | 15/8/2025 |
| **Output generated** | Yep — you can test all three discount scenarios (percentage, fixed, and inactive) directly in pgAdmin4 with some small setup queries so they work with your existing place\_orders procedure.  Here’s the plan:  **1️⃣ Percentage Discount Active**  **Setup**  -- Pick an existing member & product  SELECT id, name FROM member LIMIT 1; -- assume member\_id = 101  SELECT id, name, stock\_quantity FROM product LIMIT 1; -- assume product\_id = 7  -- Ensure product stock is high enough  UPDATE product SET stock\_quantity = 20 WHERE id = 7;  -- Remove old cart items  DELETE FROM cart\_item WHERE cart\_id = (SELECT id FROM cart WHERE member\_id = 101);  -- Ensure cart exists  INSERT INTO cart (member\_id)  SELECT 101 WHERE NOT EXISTS (SELECT 1 FROM cart WHERE member\_id = 101);  -- Add product to cart  INSERT INTO cart\_item (cart\_id, product\_id, quantity)  SELECT c.id, 7, 5 FROM cart c WHERE c.member\_id = 101;  -- Create an active percentage discount  DELETE FROM product\_discount WHERE product\_id = 7;  INSERT INTO product\_discount (  product\_id, discount\_type, value, is\_active, start\_date, end\_date, min\_quantity  ) VALUES (  7, 'percentage', 20, TRUE, NOW() - INTERVAL '1 day', NULL, NULL  );  **Test**  CALL public.place\_orders(101, true); -- Preview mode  **Expected**   * In **Messages** tab JSON, discount = 20% of unitPrice × quantity * Stock unchanged in preview mode   **2️⃣ Fixed Discount Active**  **Setup**  DELETE FROM product\_discount WHERE product\_id = 7;  INSERT INTO product\_discount (  product\_id, discount\_type, value, is\_active, start\_date, end\_date, min\_quantity  ) VALUES (  7, 'fixed', 15, TRUE, NOW() - INTERVAL '1 day', NULL, NULL  );  **Test**  CALL public.place\_orders(101, true);  **Expected**   * discount field = exactly 15 (regardless of quantity/price) * Stock unchanged in preview mode   **3️⃣ Discount Inactive**  **Setup**  DELETE FROM product\_discount WHERE product\_id = 7;  INSERT INTO product\_discount (  product\_id, discount\_type, value, is\_active, start\_date, end\_date, min\_quantity  ) VALUES (  7, 'percentage', 50, FALSE, NOW() - INTERVAL '1 day', NULL, NULL  );  **Test**  CALL public.place\_orders(101, true);  **Expected**   * discount = 0 (because is\_active = FALSE means it doesn’t apply)   **4️⃣ Expired Discount**  **Setup**  DELETE FROM product\_discount WHERE product\_id = 7;  INSERT INTO product\_discount (  product\_id, discount\_type, value, is\_active, start\_date, end\_date, min\_quantity  ) VALUES (  7, 'percentage', 50, TRUE, NOW() - INTERVAL '10 days', NOW() - INTERVAL '5 days', NULL  );  **Test**  CALL public.place\_orders(101, true);  **Expected**   * discount = 0 (because today’s date is outside discount period)   **5️⃣ Minimum Quantity Condition**  **Setup**  DELETE FROM product\_discount WHERE product\_id = 7;  INSERT INTO product\_discount (  product\_id, discount\_type, value, is\_active, start\_date, end\_date, min\_quantity  ) VALUES (  7, 'percentage', 25, TRUE, NOW() - INTERVAL '1 day', NULL, 10 -- Needs at least 10 units  );  -- Ensure cart only has 5 units  UPDATE cart\_item  SET quantity = 5  WHERE cart\_id = (SELECT id FROM cart WHERE member\_id = 101)  AND product\_id = 7;  **Test**  CALL public.place\_orders(101, true);  **Expected**   * discount = 0 because quantity < min\_quantity   If you run all 5, you’ll fully test your place\_orders discount logic without changing your stored procedure. Do you want me to also **combine these into one SQL script** so you can run them all in sequence in pgAdmin4 and compare results quickly? That would speed up your testing a lot. |
| **Impact on submission** | Helped with the report immensely when testing out different scenarios of the stored procedure as required from the report |
| **Input prompt** | help me make my cart ,cartitem,productdiscount table |
| **Output generated** | Got it — I’ve read through your CA2 brief, and I can help you add the **cart**, **cartItem**, and **productDiscount** tables into your Prisma schema.prisma so they match the requirements.  Here’s a clean, relational design you can slot into your schema:  model Cart {  cartId Int @id @default(autoincrement()) @map("cart\_id")  memberId Int @map("member\_id")  createdAt DateTime @default(now()) @map("created\_at")  updatedAt DateTime @updatedAt @map("updated\_at")  member member @relation(fields: [memberId], references: [id], onDelete: Cascade, onUpdate: NoAction)  cartItems CartItem[]  }  model CartItem {  cartItemId Int @id @default(autoincrement()) @map("cart\_item\_id")  cartId Int @map("cart\_id")  productId Int @map("product\_id")  quantity Int @default(1)  createdAt DateTime @default(now()) @map("created\_at")  updatedAt DateTime @updatedAt @map("updated\_at")  cart Cart @relation(fields: [cartId], references: [cartId], onDelete: Cascade, onUpdate: NoAction)  product product @relation(fields: [productId], references: [productId], onDelete: NoAction, onUpdate: NoAction)  }  model ProductDiscount {  discountId Int @id @default(autoincrement()) @map("discount\_id")  productId Int @map("product\_id")  discountType String @map("discount\_type") // e.g. 'PERCENTAGE', 'AMOUNT'  discountValue Decimal @map("discount\_value") @db.Decimal(10,2)  minQuantity Int? @map("min\_quantity") // For quantity-based discounts  minTotalValue Decimal? @map("min\_total\_value") @db.Decimal(10,2) // For cart-value-based discounts  startDate DateTime @map("start\_date")  endDate DateTime @map("end\_date")  product product @relation(fields: [productId], references: [productId], onDelete: Cascade, onUpdate: NoAction)  }  **Why this design works for your CA2:**   * **Cart** links directly to a member so each member only manages their own cart. * **CartItem** joins Cart and product so you can add, update, or remove specific items. * **ProductDiscount** supports both:   + Quantity-based deals (e.g., “Buy 3 get 10% off” via minQuantity).   + Total-spend deals (e.g., “Spend $100 get 5% off” via minTotalValue). * Dates in **ProductDiscount** let you handle periodic promotions. * All foreign keys have onDelete rules to keep integrity when records are removed.   If you want, I can also **link ProductDiscount to CartItem during checkout** so discounts are auto-applied when conditions are met. That would make your checkout logic easier.  Do you want me to go ahead and add that link? |
| **Impact on submission** | Helped with the creation of the schemas using prisma, also helped with general debugging |

Other source(s)

|  |  |
| --- | --- |
| **Source** | **Impact on submission** |
| < *Link to online source* > | < *Briefly explain which part of your submitted work the online source was applied* > |
| < *Author, Book / Article / Journal, Year,*  *Pages* > | < *Briefly explain which part of your submitted work the*  *Book / Article / Journal was applied* > |
| < *Student / Person name* > | < *Briefly explain which part of your submitted work the help of another student / person was applied* > |
| < *Others* > | < *Briefly explain which part of your submitted work the source was applied* > |

Singapore Polytechnic School of Computing Page **2** of **2**