# Coursework Assessment Brief

# Assessment Details

|  |  |
| --- | --- |
| Module Title: | Introduction to Databases |
| Module Code: | SWD406 |
| Module Leader: | Kenton Wheeler |
| Level: | 4 (FHEQ) |
| Assessment Title: | Database development and evaluation |
| Assessment Number: | AE1 |
| Assessment Type: | Report containing code |
| Restrictions on Time/Word Count: | Max. 1500 words for the evaluation report. Any SQL, Python code, diagrams or explanatory text for parts 1, 2 and 3 are not included in this word count. |
| Consequence of not meeting time/word count limit: | There is no penalty for submitting below the word/count limit, but students should be aware that there is a risk they may not maximise their potential mark.  Assignments should be presented appropriately in line with the restrictions stated above; if an assignment exceeds the time/word count this will be taken in account in the marks given using the assessment criteria shown. |
| Individual/Group: | Individual |
| Assessment Weighting: | 100% |
| Issue Date: | 15th February 2021 |
| Hand In Date: | 12th May 2021 |
| Mode of Submission: | On-line via SOL |
| Number of copies to be submitted: | N/A |
| Anonymous Marking | This assessment will be marked anonymously |

# Assessment Task

**1. Introduction**

This assessment requires you to undertake practical database application development work

to meet specified requirements and write an reflective evaluation report that discusses the quality of the work completed and the approaches used.

This assessment will enable students to demonstrate in full or in part the learning outcomes identified in the unit descriptor.

**2. The Practical Development Work**

The practical development work is based on an online electronics shopping company where you work as a Database Analyst/Developer. The entity-relationship diagram and SQL script for creating and populating the database are provided on SOL. You can find these and other resources required to complete the assessment on the Assessment tab.

**Part 1 - Retrieving Data using SQL**

You have been asked to write the following SQL queries for management information purposes. **All students should complete questions a, b and c below and to achieve a higher grade, also complete question d.**

1. The company want to do a marketing campaign to new shoppers and all female shoppers. Retrieve the first name, surname, email address, gender, date joined, and the current age in years of shoppers who joined on or after 1st Jan 2020 and all female shoppers (irrespective of when they joined). Print date columns in the format DD-MM-YYYY and print ‘Not known’ for any NULL values. Order results by gender and then by age (highest first).

Refer to the SQLite Built-in Functions reference on SOL for how to calculate the age and format the dates.

1. The website requires a customer account history page which will accept the shopper id as a parameter entered by the user at run time. Write a query to retrieve the first name and surname for a specific shopper along with details of all the orders they’ve placed, displaying the order no, order date, product description, seller name, quantity ordered, price (with two decimal places and prefixed by a £ sign) and ordered product status. Print date columns in the format DD-MM-YYYY. Sort the results by order date showing the most recent order first. Test your query for shopper ids 10000 and 10019.
2. The business relationship manager has asked you to write a sales summary report. Display the seller account ref, seller name, product code, product description, total quantity sold and total sales (sum of quantity\*price) for all sellers and the products they sell even if they have not sold any of a particular product (showing any NULL values as 0). Display the total sales with two decimal places and prefixed by a £ sign. Sort results by total quantity sold (lowest first).
3. The head of sales wants a report showing the products that have an average quantity sold that is less than the average quantity sold for the category that the product is in. Cancelled orders should be excluded from the calculations. Any products that haven’t sold at all should also be displayed with an average quantity of 0. Display the category description, product code, product description, average quantity sold for the product and average quantity sold for the category its in. Both averages should be displayed to an accuracy of 2 decimal places and results should be shown in category description, then product description order.

**For each query, include the SQL code you have written (in a format that can be copied and pasted i.e. not as a screenshot) along with a brief explanation of the SQL. Supply screenshots of the query results and user input (if any) and provide proof that the results are correct by doing thorough testing. Remember to display meaningful and user-friendly column headings on all queries.**

**Part 2 – Database Design, Implementation and Integrity**

The online electronics shopping database needs to be extended to store the data required to implement shopper reviews about sellers and products.

Seller reviews are just about the seller not about the product they sold and product reviews are about the product and not the seller that sold it. Each review must be star-rated as \* (Poor), \*\* (Fair), \*\*\* (Good), \*\*\*\* (Very Good) and \*\*\*\*\* (Excellent) and hold a brief textual comment from the shopper. The date and time that the feedback was submitted should also be stored.

**All students should complete questions a, b and c below and to achieve a higher grade, also complete question d.**

1. Produce a table design to support the new functionality for product and seller reviews outlined above explaining the process you used to arrive at your design, how you ensured the database integrity would be maintained and any design assumptions that you have made. **Your design should consist of at least two new tables and you must link to at least one of the existing tables.**
2. Modify the provided Orinoco entity relationship diagram to show the new entities from your design in question a, their primary and foreign keys and how they relate to each other and to the existing tables.
3. Implement your design for product and seller reviews by creating the new tables, insert at least 5 rows into each of your new tables and carry out testing to prove that your integrity constraints (primary, foreign, unique and check constraints) work correctly. Include the SQL that you used to create, populate and test the new tables.
4. Create a view that joins your newly created tables for product and/or seller reviews together at least one existing tables and provide at least two SQL queries that select from this view.

**Part 3 – Programming for Databases**

Develop Python code to implement a basic text-based application to allow the user to interact with the online electronics shopping database as outlined below. **All students should complete questions a, b and c below and to achieve a higher grade, also complete question d.**

1. i. Prompt for the entry of a shopper\_id which will be used to test all the menu options. If the shopper\_id entered is found, print a welcome message including the name of the shopper. If the shopper\_id is not found in the database, print an error message and exit the program otherwise print the main menu below.

ii. Print a text-based menu as follows:

ORINOCO – SHOPPER MAIN MENU

1. Display your order history
2. Add an item to your basket
3. View your basket
4. Change the quantity of an item in your basket
5. Remove an item from your basket
6. Checkout
7. Exit

iii. As shoppers should be able to resume a basket previously created from a previous execution of the program on the same day, check if there is a row in the shopper\_baskets table created today for the selected shopper and, if so, make this the current basket otherwise create a new (empty) basket. If there is more than one basket created today for the shopper, use the most recent one.

You can use the following SQL query to return the most recent basket for the current shopper created today (if there is one):

SELECT basket\_id

FROM shopper\_baskets

WHERE shopper\_id = ?

AND DATE(basket\_created\_date\_time) = DATE('now')

ORDER BY basket\_created\_date\_time DESC

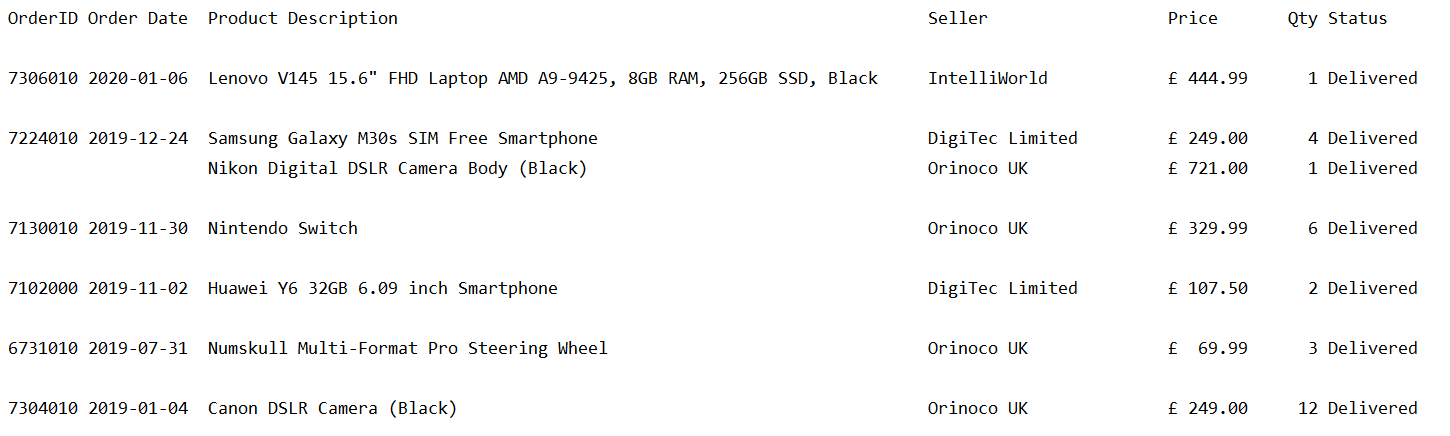
LIMIT 1

When you execute the query, pass the shopper\_id as a parameter to replace the ? placeholder.

1. Implement menu options 1 and 7 as follows:

Option 1 – Display your order history

1. For each order that the customer has placed, display the order id and order date together with the product description, seller name, price, quantity ordered and status of each product on that order. You can use the query you wrote for Question 1b of this assessment as a basis for the SQL query for this option.
2. Sort orders by order date (most recent first)
3. If no orders are found for the shopper\_id that you are testing with, print the message “No orders placed by this customer”
4. Display the data in the format shown below (which is for shopper\_id 10010)



1. Return to the main menu

Option 7 – Exit

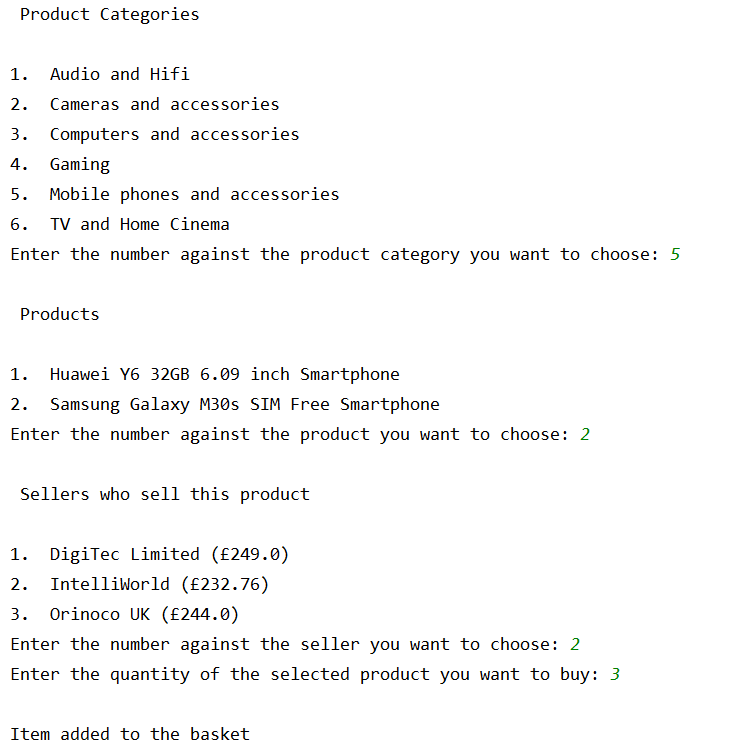
1. Exit the program
2. Implement menu options 2 and 3 as follows:

Please note: The details of the shopper’s basket should be stored in the shopper\_baskets and basket\_contents tables and not in a Python data structure (like a list). This will allow a shopper to continue with their last basket if they didn’t complete the checkout in a previous execution of the program.

Option 2 – Add an item to your basket

1. Display a numbered list of product categories
2. Prompt the user to enter the number of the product category they want to choose from and store the category\_id for the selected category
3. Display a numbered list of the available products in the category selected
4. Prompt the user to enter the number of the product they want to purchase and store the product\_id for the selected product
5. Display a numbered list of sellers who sell the product they have selected and the price they are selling that product at
6. Prompt the user to enter the seller they wish to buy the product from and store the seller\_id for the selected seller
7. Prompt the user to enter the quantity of the selected product they want to order. Display ‘The quantity must be greater than 0’ if the quantity is <=0 and re-prompt the user to enter it again.
8. Get the price of the selected product from the selected supplier
9. If the basket is empty, get the next basket id by selecting from the sqlite\_sequence table and insert a new row into the shopper\_baskets table using the next basket \_id.
10. Insert a new row into the basket\_contents table for the product they’ve chosen to purchase using the basket id selected in stage ix. All items added to the basket should have the same basket\_id in the basket\_contents table.
11. Commit the transaction
12. Print “Item added to your basket”
13. Return to the main menu

Below is an example of what should be displayed and what should be prompted for:



To simplify your code, the following function can be included at the top of your program to display a numbered list of options and return the id of the selected option. You can amend this function accordingly if you wish but a function must be included.

**def** \_display\_options(all\_options,title,type):  
 option\_num = 1  
 option\_list = []  
 print(**"\n"**,title,**"\n"**)  
 **for** option **in** all\_options:  
 code = option[0]  
 desc = option[1]  
 print(**"{0}.\t{1}"**.format(option\_num, desc))  
 option\_num = option\_num + 1  
 option\_list.append(code)  
 selected\_option = 0  
 **while** selected\_option > len(option\_list) or selected\_option == 0:  
 prompt = **"Enter the number against the "**+type+**" you want to choose: "** selected\_option = int(input(prompt))  
 **return** option\_list[selected\_option - 1]

This function should be called in steps i, iii and v above using a command of the following format:

id\_of\_selected\_option = \_display\_options(query\_rows,**title**,**type**)

query\_rows must consist of two values - id and description i.e. the category\_id and category\_description

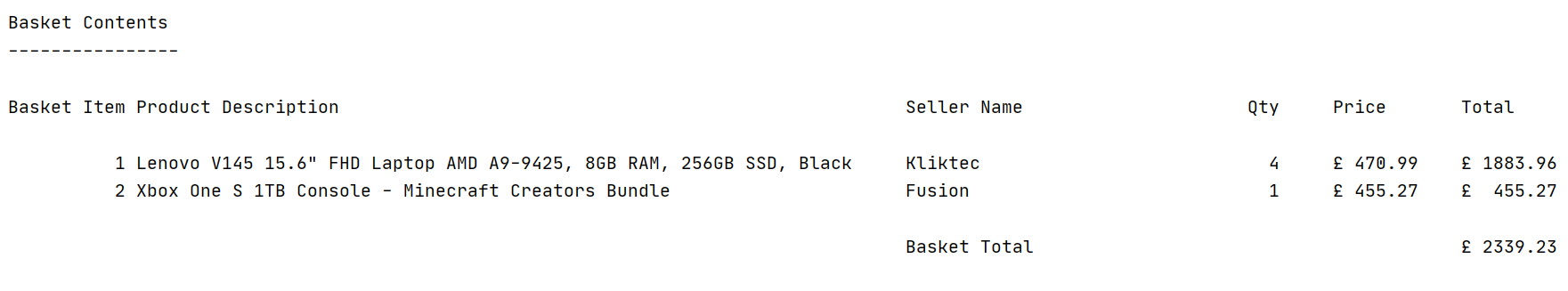
title is some text to put above the list of options to act as a title

type is used to customise the prompt to make it appropriate for what you want the user to select

Option 3 – Display your basket

1. If the basket is empty, display ‘Your basket is empty’ otherwise display all rows from the basket\_contents table for the current basket, labelling each item with a basket item no. starting at 1. Also display a total basket cost.

An example of how the basket should be displayed is shown below:



1. Return to the main menu
2. Implement the remaining menu options (4, 5 and 6) as follows:

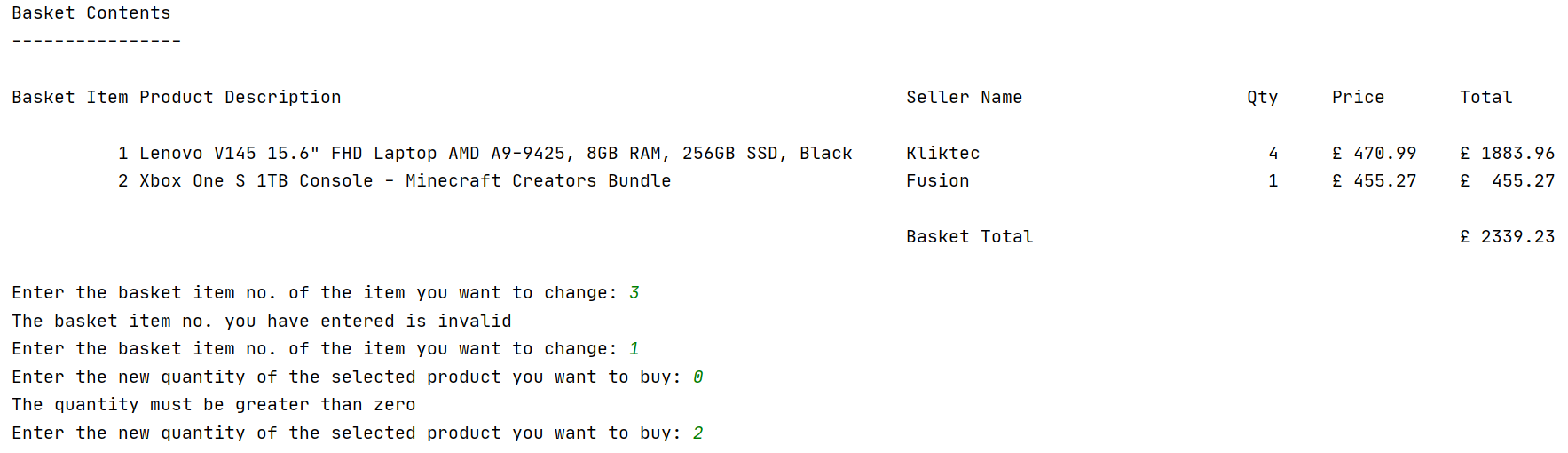
Option 4 – Change the quantity of an item in your basket

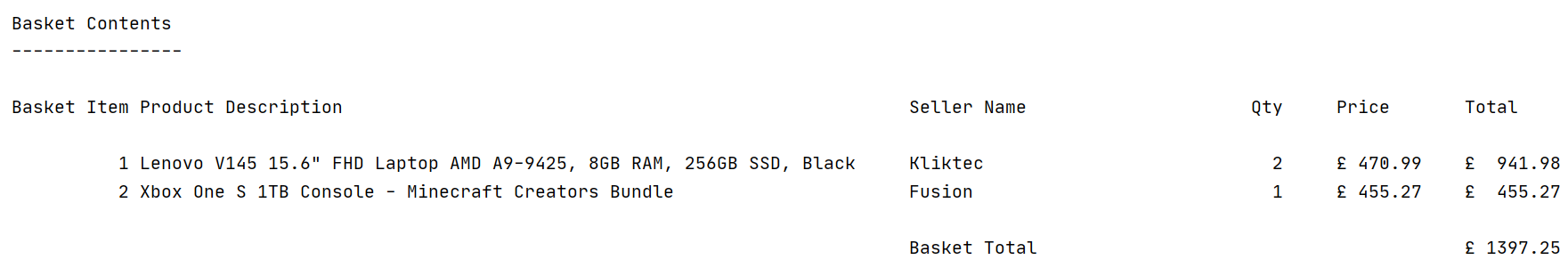
1. If the basket is empty, display ‘Your basket is empty’ and return to the main menu otherwise display the current basket and the basket total (as per option 3.
2. If there is more than one item in the basket, prompt the user to enter the basket item no. of the item they want to update. If they enter an invalid basket item no., display ‘The basket item no. you have entered is invalid’ and re-prompt the user to enter it again.

If there is only one item in the basket, this will obviously be the one the user wants to change.

1. Prompt the user to enter the new quantity for the item selected. If they enter a quantity <= 0, display ‘The quantity must be greater than 0’ and re-prompt the user to enter it again.
2. Update the basket\_contents table with the new quantity for the current basket and item that has been changed.
3. Display the current basket with a re-calculated total.
4. Return to the main menu

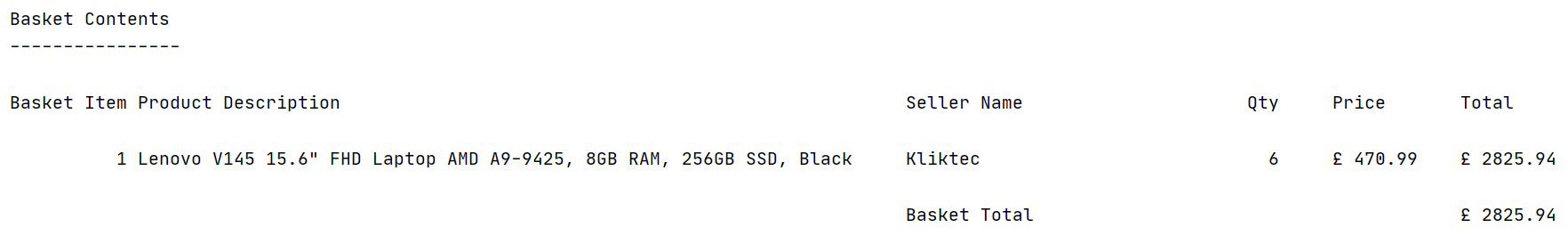
Below is an example of what should be displayed and what should be prompted if there is more than one item in the basket:





Below is an example of what should be displayed and what should be prompted if there is only one item in the basket:



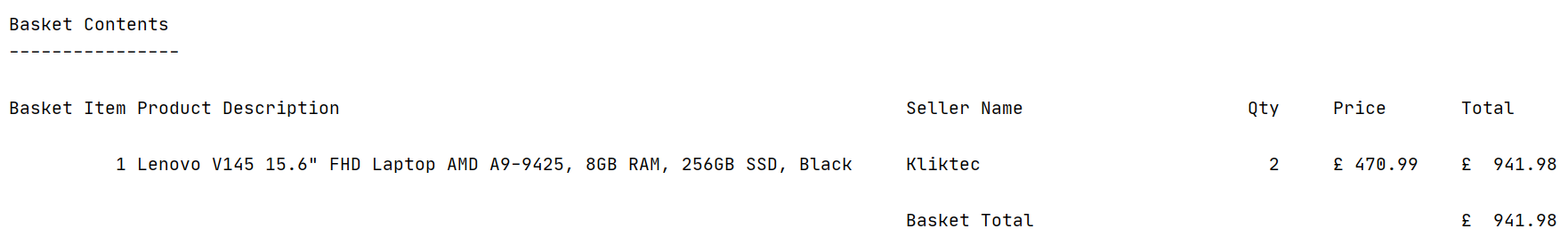


Option 5 – Remove an item from your basket

1. If the basket is empty, display ‘Your basket is empty’ otherwise display the current basket and the basket total as per option 3.
2. Prompt the user to enter the product id of the basket item they want to remove. If the product id entered is not in the current basket, display ‘The product id you have entered is not in your basket’
3. Prompt the user to confirm they definitely want to remove the selected item from their basket by entering Y or N.
4. If the user confirms they definitely want to remove the selected item, delete the item from the current basket in the basket\_contents table.
5. Check if the basket is now empty and if so, delete the row from the shopper\_baskets table for the current basket and display ‘Your basket is empty’ otherwise display the current basket with a re-calculated total.
6. Return to the main menu

Below is an example of what should be displayed and what should be prompted if there is more than one item in the basket:





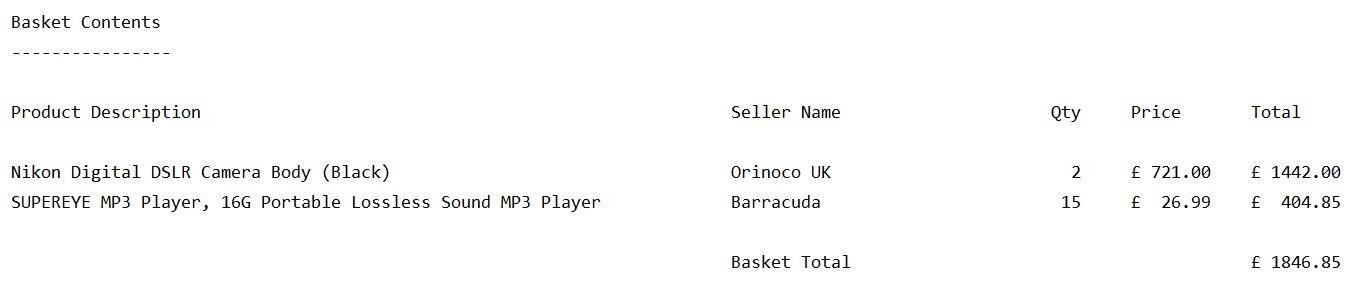
Below is an example of what should be displayed and what should be prompted if there is only one item in the basket:



Option 6 – Checkout your basket

1. If the basket is empty, display a suitable message and return to the main menu
2. Display the current basket and the basket total (the same as option 3) and ask the user if they wish to proceed with the checkout (Y or N). If they enter N, return to the main menu. If they enter Y, continue as follows:
3. Insert a new row into the shopper\_order table for the basket with a status of ‘Placed’
4. Insert a new row into the ordered\_product table for each item in the basket with a status of ‘Placed’
5. Delete the rows from the shopper\_basket and basket\_contents tables for this basket
6. Print the message ‘Checkout complete, your order has been placed’
7. Return to the main menu

Below is an example of what should be displayed for a shopper who proceeds with the checkout process:



Do you wish to proceed with the checkout (Y or N)? Y

Checkout complete, your order has been placed.

You should add comments throughout your code to make it easier for someone else to understand.

**With your submission, you must include all your Python code, screenshots of the output and any user interaction together with evidence that the requirements outlined under each menu option have been met and thoroughly tested by including screenshots of data successfully inserted, updated and deleted from the database and errors/exceptions being correctly handled.**

**3. Evaluation Report (guideline - approx 1500 words)**

**All students must include a report in their submission which should cover the following:**

a) An evaluation of the quality of the work you have produced with reference to both the specified requirements and the assessment criteria. You should include:

* any deliverables that are not complete or fully tested;
* elements of the practical work that you feel contribute to higher grade achievement;
* your approach to design and testing of the SQL and Python elements of the assessment;
* how well you feel you managed your time between the four parts of the assessment and what more you would have done if you’d had more time.

b) Bibliography and references

**4. Submitting Your Work**

The online submission (which should be well structured and clearly written) must be in a single document in MS Word or PDF format and should include:

**1.** A title page

**2.** Contents page and page numbers.

**3.** Copies of all your **final (versions** of the deliverables for parts 1, 2 and 3.

**4.** Your report (see section 3 above)

# Assessment criteria

The summary grid below is the basis for grading achievement. Higher levels of achievement are described towards the right-hand side of the grid. Each **level** subsumes the previous level. An indication of the % contribution of each part and the report is indicated. Normally, an attempt must be made for each block and the report.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S, F3-F1** | **D3-D1** | **C3->C1** | **B3->B1** | **A4->A1** |
| Part 1 - Retrieving Data using SQL – Learning Outcomes K1, C1, P1 – 30% | | | | |
| Does not reach required threshold. | At least two queries were submitted that produce the correct or mostly correct results.  The queries meet the basic requirements but no attempt has been made to improve the presentation.  No evidence of testing has been provided to ensure the results for each query are correct.  No explanation has been provided for each query. | Has submitted queries a, b and c and they all produce the correct or mostly correct results.  The queries meet the basic requirements and some attempt has been made to improve the presentation.  Some evidence of testing has been provided for some of the queries but the testing is poor.  Some explanation has been provided for each query demonstrating a basic understanding of the concepts. | Has submitted queries a, b and c and they all produce the correct results  The queries meet the majority of the requirements and the output is well presented.  Some evidence of testing has been provided to ensure the results for each query are correct.  Some explanation has been provided for each query demonstrating a good understanding of the concepts. | Has submitted queries a, b,c and d and they all produce the correct results.  The queries meet or exceed all the requirements and the output is professionally presented.  Evidence of through testing has been provided to ensure the results for each query are correct.  A detailed explanation has been provided for each query demonstrating a full understanding of the concepts. |
| Part 2 – Database Design Implementation and Integrity - Learning Outcomes K1, C1, P1 – 30% | | | | |
| Does not reach required threshold. | The design has some flaws and only partially meet the requirements.  The design has a minimum of two new tables and links to at least one existing table.  The rationale behind the design has not been provided and any design assumptions have not been outlined.  The newly-created tables have not been correctly implemented or the primary and foreign keys are missing or incorrect.  Each new table has not been populated with sufficient test data.  No evidence has been provided that the new constraints have been tested to ensure they work correctly.  No amended ERD has been provided or the one provided is incorrect.  No view has been created or used. | The design has some flaws but mostly meets the requirements  The design has a minimum of two new tables and links to at least one existing table.  A basic rationale behind the design has been provided but no design assumptions have been outlined.  The newly-created tables have been correctly implemented with the correct primary and foreign keys. No other constraints have been used  Each new table has been populated with basic test data.  Evidence has been provided that some of the new constraints have been tested to ensure they work correctly.  An amended ERD diagram has been provided but it is incorrect.  No view has been created or used. | The design is correct, and meets the requirements.  The design has a minimum of three new tables and links to at least two existing tables.  The rationale behind the design has been explained and any design assumptions have been outlined.  The newly-created tables have been correctly implemented with the right primary and foreign keys. Some other constraints have been used.  Each new table has been populated with a reasonable amount of test data.  Evidence has been provided that some of the new constraints have been tested to ensure they work correctly.  A correctly amended ERD diagram has been provided.  If a view was created, it was not fully tested with at least two correct queries. | The design is correct and meets or exceeds the requirements  The design has a minimum of three tables and links to at least two existing tables.  The rationale behind the design has been fully explained and any design assumptions have been outlined.  The newly-created tables have been correctly implemented with the right primary and foreign keys. Other constraints have been widely used and the constraints have been named.  Each new table has been populated with a good amount of test data.  Full evidence has been provided that all new constraints have been tested to ensure they work correctly.  A correctly amended ERD diagram has been provided.  A view has been created based on the new and existing tables and at least two queries correctly written using the view. |
| Part 3 - Programming Learning Outcomes K1 and T1 – 30% | | | | |
| Does not reach required threshold. | Menu option 1 has been attempted but it does not work correctly or does not fully meet the requirements.  The user interface is basic or does not function correctly.  The code retrieves data from the database  The program has no functions.  There are some major issues with how the code has been written.  There is little or no evidence that the code has been tested  Comments are not used in the code  There is no evidence of error handling. | Menu option 1 has been fully implemented, and works correctly and menu options 2 and 3 are partially implemented or do not function correctly.  The user interface is basic but functional.  The code retrieves data from the database  The program uses the supplied function without modification.  There are some minor issues with how the code has been written.  There is some evidence that the code has been tested  Comments are sparsely used in the code  There is some evidence of error handling. | Menu options 1,2 and 3 have been fully implemented, work correctly and mostly meet the requirements. Menu options 4, 5 or 6 have been attempted but are only partially implemented or do not work correctly.  The user interface is good with clear prompts and helpful error messages.  The code retrieves data from the database and inserts rows in the database.  The program uses the supplied function or a modified version of it.  The code is correctly structured and well-written.  There is evidence that the code has been tested  Clear comments are used throughout the code.  There is some evidence of error handling. | All menu options have been fully implemented, they all work correctly and fully meet the requirements.  The user interface is good with clear prompts and helpful error messages.  The code retrieves data from the database and inserts, updates and deletes rows in the database.  The program makes extensive use of functions to modularise the code.  The code is correctly structured and well-written.  There is evidence that the code has been thoroughly tested  Clear comments are used throughout the code.  Extensive error handling has been used. |
| Evaluation Report = 10% | | | | |
| Does not reach required threshold. | A basic description of the approach you used to produce your deliverables and evaluation of the work you have produced. | Some reflection on the approach you used to produce your deliverables and the quality of the work you have produced. | An honest self-assessment of the approach you used to produce the deliverables and the quality of the work you have produced stating what you did well and not so well. | An honest, reflective and insightful evaluation of the approach you used to produce the deliverables and the quality of the work you have produced stating what you did well and not so well and what more you would have done if you’d had more time. |