# Solent University

# Coursework Assessment Brief

# Assessment Details

|  |  |
| --- | --- |
| Module Title: | Introduction to Databases |
| Module Code: | COM417 |
| Module Leader: | Kenton Wheeler |
| Level: | 4 (FHEQ) |
| Assessment Title: | Database queries, design and development |
| Assessment Number: | AE1 |
| Assessment Type: | Report containing code |
| Restrictions on Time/Word Count: | Max. 1500 words for any explanations or narrative written for the three practical parts. Any SQL and Python code is 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: | 12th February 2024 |
| Hand In Date: | 3rd May 2024 |
| Planned Feedback Date: | 31st May 2024 |
| Mode of Submission: | On-line via SOL  **Only FINAL submissions will be accepted. DRAFT submissions will not be considered an attempt and will not be marked.** |
| Number of copies to be submitted: | N/A |
| Anonymous Marking | This assessment will be marked anonymously. **Do not include your name, only your student ID.** |

# Assessment Task

**1. Introduction**

This assessment requires you to undertake practical database application development work

to meet specified requirements.

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

**PLEASE NOTE: Students must use the tables and data supplied in the Paraná create script (available on the Assessments tab on SOL) to answer parts 1, 2 and 3 of this assessment.**

**Part 1 - Retrieving Data using SQL (worth 35% of the marks)**

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.

The output below is shown to illustrate the format of the results that your query should produce, not the actual results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Shopper first name | Shopper surname | Email address | Gender | Date Joined | Current Age |
| Jane | Doe | Jane.doe@gmail.com | F | 15-06-2019 | 56 |
| Rachel | Doe | rdoe1565@hotmail.co.uk | M | 30-11-2020 | 47 |
| Kris | Doe | krisdoe@yahoo.co.uk | Not known | 12-10-2020 | 22 |

1. The website requires a shopper order history page which will accept the shopper id as a parameter entered by the user at run time and will display the order history for only that one shopper. Write a query to retrieve the first name and surname for a specific shopper along with details of all the orders they’ve made, displaying the order id, 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 by running it twice once for shopper id 10000 and a second time for shopper id 10019.

The output below is shown to illustrate the format of the results that your query should produce, not the actual results

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Shopper first name | Shopper surname | Order ID | Order Date | Product Description | Seller Name | Qty  ordered | Price | Order Status |
| Jane | Doe | 1234 | 02-03-2020 | Sony Bravia KD43 TV | ABC Ltd | 1 | £419.99 | Delivered |
| Jane | Doe | 1234 | 02-03-2020 | Apple iPhone 13 | ABC Ltd | 2 | £699.99 | Delivered |
| Jane | Doe | 1345 | 08-03-2020 | HP Deskject 2700 | Colorado | 5 | £57.95 | Dispatched |

1. The business relationship manager has asked you to write a sales summary report. Display the seller account ref, seller name, product code, product description, number of orders, total quantity sold and total sales (calculated as the sum of quantity\*price) for all sellers and the products they sell even if they have not sold any of a particular product. NULL values should be displayed as 0, display the total sales with two decimal places and prefixed by a £ sign and sort results by total quantity sold (lowest first).

The output below is shown to illustrate the format of the results that your query should produce, not the actual results

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Seller Account Ref | Seller Name | Product Code | Product Description | No. of Orders | Total quantity sold | Total Value of Sales |
| COL0799 | Colorado | 304409 | Sony Bravia KD43 TV | 0 | 0 | £0.00 |
| ABC0055 | ABC Ltd | 305504 | Apple iPhone 13 | 2 | 4 | £2799.96 |
| ABC0055 | ABC Ltd | 304409 | Sony Bravia KD43 TV | 7 | 7 | £2939.93 |
| COL0799 | Colarado | 306050 | HP Deskjet 2700 | 6 | 10 | £579.50 |

1. The head of sales wants a summary 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 sort the results by category description and then product description.

The output below is shown to illustrate the format of the results that your query should produce, not the actual results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category Description | Product Code | Product Description | Avg Qty Sold | Avg Qty Sold for Category |
| Mobile Phones | 378292 | Nokia X20 | 2.56 | 7.55 |
| Printers | 310383 | HP LaserJet 3199 | 4.15 | 4.56 |
| TV & Audio | 308893 | Digihome 43552UHDH LED TV | 0 | 4.10 |
| TV & Audio | 390292 | LG 4K Ultra HD TV | 3.50 | 4.10 |

**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 (worth 30% of the marks)**

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.

Your design should allow multiple reviews to be stored about a particular product or a particular seller and multiple reviews to be submitted by a specific shopper.

**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. You should not amend the existing tables in any way such as adding new columns or foreign keys.**
2. Modify the provided Paraná 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 and insert enough rows into all of your new tables in order to test that all the functional requirements are met. Include the SQL that you used to create, populate and test the new tables.
4. Carry out additional testing to prove that your integrity constraints (primary, foreign, unique, not null and check constraints) work correctly. Include the SQL commands that you used to perform the tests and the screenshot of the error produced.

**Part 3 – Programming for Databases (worth 35% of the marks)**

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:

PARANÁ – 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. 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.

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

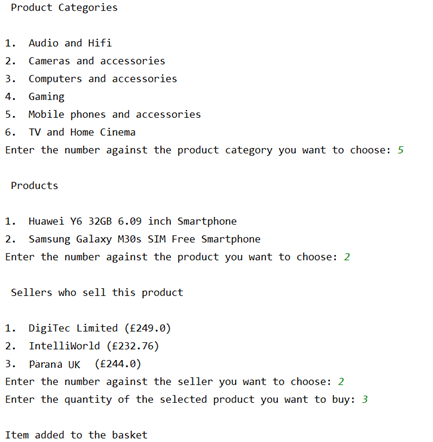
1. Exit the program
2. Implement menu options 2 and 3 as follows:

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 in alphabetical order
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 in alphabetical order.
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 in alphabetical order by seller name.
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 there is no current basket, 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

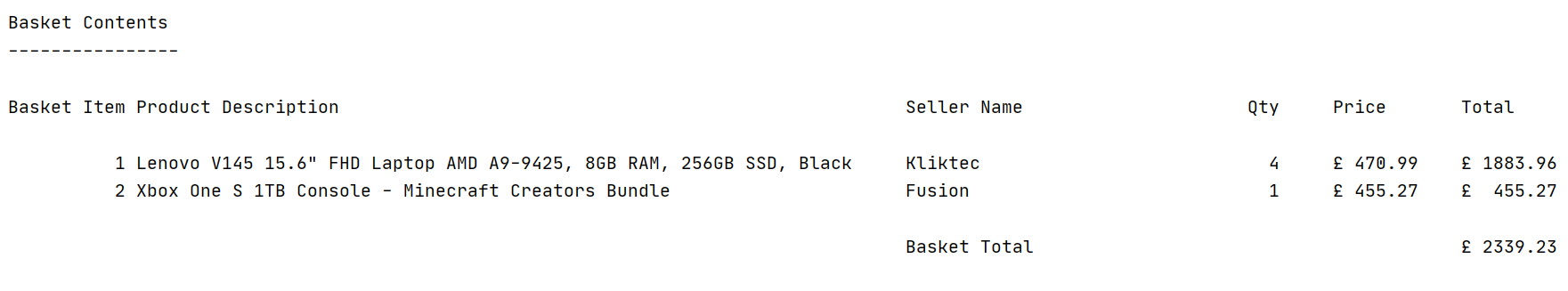
title is some text to put above the list of options to act as a title

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 and 5 as follows:

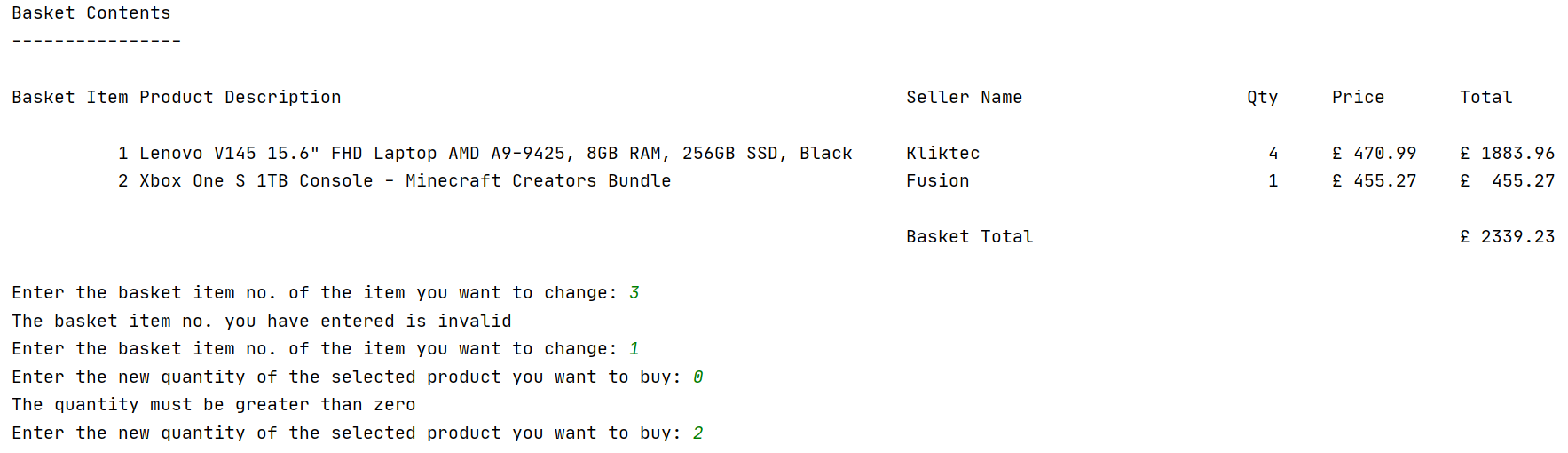
Option 4 – Change the quantity of an item in your basket

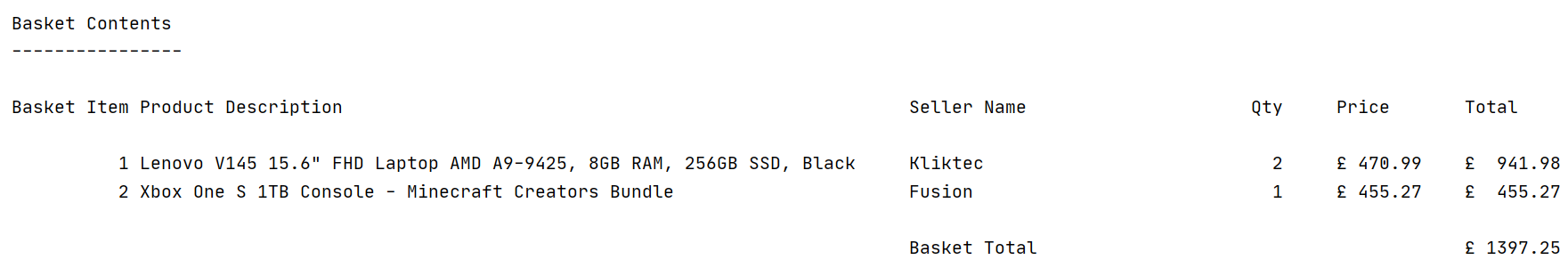
1. If the basket is empty, display ‘Your basket is empty’ and return to the main menu otherwise display the current basket and the basket total (as per option 3.
2. If there is more than one item in the basket, prompt the user to enter the basket item no. of the item they want to update. If they enter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

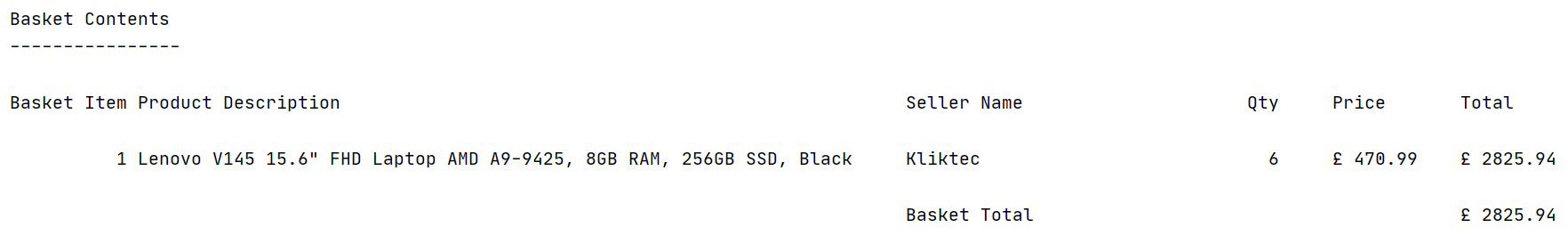
Below is an example of what should be displayed and what should be prompted if there is more than one item in the basket:





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

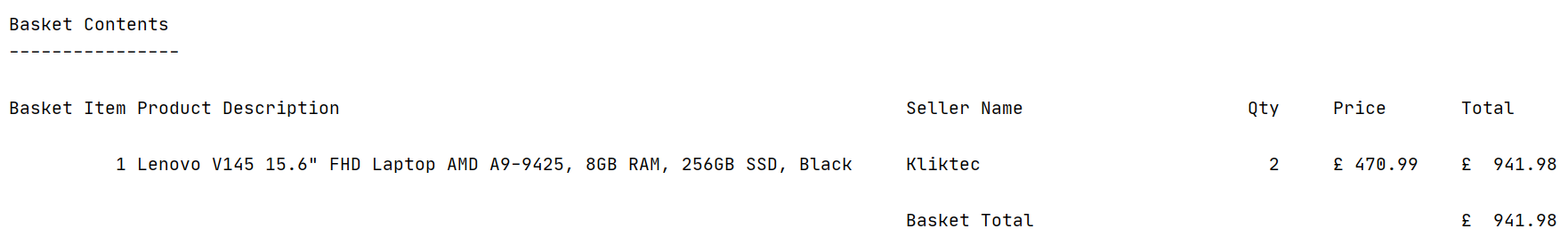
1. If the basket is empty, display ‘Your basket is empty’ otherwise display the current basket and the basket total as per option 3.
2. If there is more than one item in the basket, prompt the user to enter the basket item no. of the item they want to remove. If they enter an invalid basket item no., 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 remove.

1. Prompt the user to confirm they definitely want to remove the selected item from their basket by entering Y or N.
2. If the user confirms they definitely want to remove the selected item, delete the item from the current basket in the basket\_contents table.
3. Check if the basket is now empty and if so, display ‘Your basket is empty’ otherwise 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:



1. Implement menu option 6 as follows:

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\_orders table for the basket with a status of ‘Placed’
4. Insert a new row into the ordered\_products table for each item in the basket with a status of ‘Placed’
5. Delete the rows from the basket\_contents and shopper\_baskets 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. 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:

**a.** A title page

**b.** Contents page and page numbers.

**c.** Answers to the questions in parts 1, 2 and 3

**A template submission document is provided on SOL if you wish to use it.**

# 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. An indication of the % contribution of each part of the assessment is also provided below. Students are encouraged to attempt all parts.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Standard** | **Fails to meet expectations for all learning outcomes** | **Meets expectations for all learning outcomes** | **Exceeds expectations for some learning outcomes and meets expectations for the rest** | **Exceeds expectations for many learning outcomes and meets expectations for others** | **Exceeds expectations for all learning outcomes** |
| **Solent Grade** | **S, F3-F1** | **D3-D1** | **C3->C1** | **B3->B1** | **A4->A1** |
| **Part 1 - Retrieving Data using SQL**  **(Worth 35%)** | Does not reach required threshold. | Queries a, b and c were attempted. Only one query produces the correct results and the others are incorrect.  Little or no attempt has been made to improve the formatting or presentation of results.  No best practice has been followed in writing the SQL queries so the SQL code is difficult to read.  Some explanation has been provided for each query but it is inaccurate or uses the wrong terminology.  No evidence of testing has been provided to ensure the results for each query are correct. | Queries a, b and c were attempted. Two queries produce correct results and the other produces incorrect results.  The results have been mostly formatted correctly but the column headings have not been changed.  Some best practice has been followed in writing the SQL queries but the code is not easy to read.  A brief explanation has been provided for each query using the right terminology demonstrating a basic understanding of the concepts.  Some evidence of testing has been provided for some of the queries but the testing is insufficient. | Queries a, b and c were attempted and all three produce the correct or mostly correct results.  The results have been formatted correctly and some column aliases have been used to improve the presentation.    Best practice has been mostly followed in writing the SQL queries making them easy to read.  An accurate explanation has been provided for each query demonstrating a good understanding of the concepts.  Reasonable evidence of testing has been provided for most queries to ensure the results are correct. | Queries a, b and c were attempted and all produce the correct results.  The results have been formatted correctly and column aliases have been used throughout to improve the presentation.  All best practice has been followed in writing the SQL queries making them easy to read and understand.  A detailed explanation has been provided for each query demonstrating a full understanding of the concepts.  Evidence of through testing has been provided for all queries to ensure the results for each one are correct. |
| Query d was not attempted | Query d was attempted but the results are completely incorrect. | A reasonable attempt was made at query d and the results are partially correct. | Query d was attempted and produces the correct or mostly correct results |
| **Part 2 – Database Design Implementation and Integrity**  **(Worth 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.  No amended ERD has been provided or the one provided is very poor  The newly-created tables have not been correctly created or the primary and foreign keys are missing or incorrect.  Each new table has not been populated with sufficient data to test any of the functional requirements. | The design has some flaws but mostly meets the requirements.  The design has a minimum of two new tables and links to at least two existing tables.  A basic rationale behind the design has been provided but no design assumptions have been outlined.  An amended ERD diagram has been provided but it is incorrect or incomplete  The newly-created tables have been correctly created with the correct primary and foreign keys. No other constraints have been used.  Each new table has been populated with sufficient data to test some of the functional requirements | 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.  An amended ERD diagram has been provided and it is mostly correct.  The newly-created tables have been correctly created with the right primary and foreign keys. Some other constraints have been used.  Each new table has been populated with a good amount of data to test most of the functional requirements | 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 minimising duplication of columns between the new entities.  The rationale behind the design has been fully explained and any design assumptions have been outlined.  A correctly amended ERD diagram has been provided.  The newly-created tables have been correctly created 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 to extensively test all the functional requirements |
| No evidence has been provided that the new constraints have been tested. | Evidence has been provided that a few of the new constraints have been tested to ensure they work correctly. | Evidence has been provided that most of the new constraints have been tested to ensure they work correctly. | Full evidence has been provided that all new constraints have been tested to ensure they work correctly. |
| **Part 3 – Programming for Databases**  **(Worth 35%)** | Does not reach required threshold. | The menu and menu options 1 and 7 have been fully implemented, they function correctly and mostly meet the requirements. Any other menu options that have been attempted are only partially implemented so do not function correctly or meet the requirements.  The user interface is basic or does not function correctly.  The code retrieves data from the database  The program uses 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 or exception handling. | The menu and at least menu options 1,2,3 and 7 have been fully implemented, they function correctly and mostly meet the requirements.  Any other menu options that have been attempted are only partially implemented so 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 or only one other function.  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 but no exception handling | The menu and menu options 1,2,3 and 7 have been fully implemented, they function correctly and mostly meet the requirements. Menu options 4 and 5 have been fully implemented but do not function correctly or fully meet the requirements  The user interface is generally 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) and some other functions to modularise the code.  The code is mostly correctly structured and well-written.  There is evidence that the code has been fully tested but little/no evidence that the data has been successfully inserted, updated or deleted correctly in the database  Clear comments are used throughout the code.  There is some evidence of error and exception handling. | The menu and menu options 1,2,3,4,5 and 7 have been fully implemented, they all work correctly and fully meet the requirements.  The user interface is excellent 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 and that the data has been successfully inserted, updated and deleted in the database.  Extensive comments are used throughout the code.  Extensive error and exception handling has been used in the code. |
| Menu option 6 has not been implemented | Menu option 6 has only been partially implemented so does not function correctly or meet any of the requirements | Menu option 6 has been fully implemented but does not function correctly or only partially meets the requirements | Menu option 6 has been fully implemented, it functions correctly and fully meets the requirements. |

# Learning Outcomes

This assessment will enable you to demonstrate in full or in part your fulfilment of the following learning outcomes identified in the Module Descriptor:

**Living CV**

As part of the University's Work Ready, Future Ready strategy, you will be expected to build a professional, Living CV as you successfully engage and pass each module of your degree.

The Living CV outputs evidenced on completion of this assessment are:

1. Understand the importance and role of relational databases in modern IT systems

2. Have a good understanding of the SQL language and be able to write a range of SQL queries to meet specific reporting requirements

3. Able to design, implement and test relational databases to maintain the integrity of the data

4. Able to develop applications that securely interact with a backend database

Please add these to your CV via the Living CV builder platform on Solent Futures Online [Solent Futures Online](https://eur03.safelinks.protection.outlook.com/?url=https%3A%2F%2Fsolentfutures.careercentre.me%2Fprogrammes%2F%3FprogrammeID%3DThzJ%252bRbk%252bQXoSlEaujPR0g%253d%253d&data=04%7C01%7Cian.harris%40solent.ac.uk%7Cf1bda34c4d564e82f6cb08da067fdf48%7Cd684e4cd491a4577bf33546478d72e3c%7C0%7C0%7C637829443517919744%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000&sdata=ObCFbM3zY7CgU6SVNtitaq1udg0%2Bzlp1GuCAJ1y1utw%3D&reserved=0)

# Late Submissions

You are reminded that:

1. If this assessment is submitted late i.e. within 7 calendar days of the submission deadline, the mark will be capped at 40% if a pass mark is achieved;
2. If this assessment is submitted later than 7 calendar days after the submission deadline, the work will be regarded as a non-submission and will be awarded a zero;
3. If this assessment is being submitted as a referred piece of work, then it must be submitted by the deadline date; any Refer assessment submitted late will be regarded as a non-submission and will be awarded a zero.

[Assessment regulations](https://www.solent.ac.uk/about/documents/assessment-regulations.pdf)

# Extenuating Circumstances

The University’s Extenuating Circumstances (EC) procedure is in place if there are genuine short term exceptional circumstances that may prevent you submitting an assessment. If you are not 'fit to study’, you can either request an extension to the submission deadline of 7 calendar days or you can request to submit the assessment at the next opportunity, i.e. the resit period (as a Defer without capping of the grade). In both instances you must submit an EC application with relevant evidence. If accepted under the university regulations there will be no academic penalty for late submission or non-submission dependent on what is requested. You are reminded that EC covers only short term issues (20 working days) and that if you experience longer term matters that impact on your learning then you must contact the Student Hub for advice.

Please find a link to the EC policy below:

[Extenuating Circumstances](https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/2p-extenuating-circumstances.pdf)

# Academic Misconduct

Any submission must be your own work and, where facts or ideas have been used from other sources, these sources must be appropriately referenced. The University’s Academic Handbook includes the definitions of all practices that will be deemed to constitute academic misconduct. You should check this link before submitting your work.

Procedures relating to student academic misconduct are given below:

[Academic Misconduct](https://students.solent.ac.uk/official-documents/quality-management/academic-handbook/4l-student-academic-misconduct-procedure.pdf)

**Ethics Policy**

The work being carried out must be in compliance with the university Ethics Policy. Where there is an ethical issue, as specified within the Ethics Policy, then you will need an ethics release or ethics approval prior to the start of the project.

The Ethics Policy is contained within Section 2S of the Academic Handbook:

[Ethics Policy](https://staff.solent.ac.uk/official-documents/quality-management/academic-handbook/2s-solent-university-ethics-policy.pdf)

**Grade marking**

The University uses an alpha numeric grade scale for the marking of assessments. Unless you have been specifically informed otherwise your marked assignment will be awarded a letter/number grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

[Grade Marking Scale](https://staff.solent.ac.uk/official-documents/quality-management/academic-handbook/2o-assessment-regulations-annex-1-grade-marking-scale.pdf)

**Guidance for online submission through Solent Online Learning (SOL)**

[Online Submission](http://learn.solent.ac.uk/onlinesubmission)