SWD406 AE1 report

1.Introduction

2.Practical Development Work

1. Retrieving data using SQL

a)

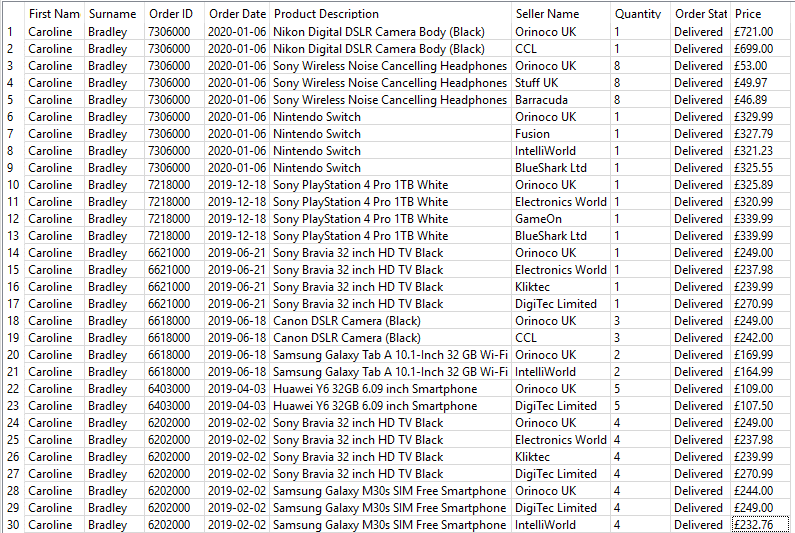
1. **SELECT**
2. Shopper\_first\_name **AS** [**first** **name**]
3. ,Shopper\_surname **AS** [surname]
4. ,Shopper\_email\_address **AS** [email address]
5. ,(strftime('%Y', 'now') - strftime('%Y', Date\_of\_birth) ) - (strftime('%m-%d', 'now') < strftime('%m-%d', Date\_of\_Birth) ) **AS** [age]
6. ,date\_joined **AS** [**date** joined]
7. **FROM** Shoppers
8. **WHERE** Date\_joined >= '2020-01-01' OR Date\_of\_birth > '1990-01-01'
9. **ORDER** **BY** Date\_of\_birth, Shopper\_surname
10. ;



* firstly, I selected the necessary data and named the respective columns appropriately, using the SELECT function, then subsequently the AS function for the naming.
* The date of birth was ‘converted’ to age by subtracting the year from the date of birth, from the current year; this is then displayed as ‘age’, again using the AS function.
* Then the table it should be drawn from is declared using the FROM function.
* Using the WHERE function, the results are filtered to only include those who joined after 2019 or were 29 and older.
* Finally, the results are ordered first by age, then by surname alphabetically.

b)

1. **SELECT**
2. s.shopper\_first\_name **AS** [**First** **Name**]
3. ,s.shopper\_surname **AS** [Surname]
4. ,so.Order\_id **AS** [**Order** ID]
5. ,so.Order\_date **AS** [**Order** **Date**]
6. ,p.product\_description **AS** [Product Description]
7. ,sell.Seller\_name **AS** [Seller **Name**]
8. ,op.Quantity **AS** [Quantity]
9. ,op.Ordered\_product\_status **AS** [**Order** Status]
10. ,PRINTF("£%.2f",ps.Price) **AS** [Price]
11. **FROM** shopper\_orders so
12. **INNER** JOIN
13. shoppers s **ON** s.shopper\_id = so.shopper\_id
14. **INNER** JOIN
15. ordered\_products op **ON** op.order\_id = so.order\_id
16. **INNER** JOIN
17. products p **ON** p.product\_id = op.product\_id
18. **INNER** JOIN
19. product\_sellers ps **ON** ps.product\_id = op.product\_id
20. **INNER** JOIN
21. sellers sell **ON** sell.seller\_id = ps.seller\_id
22. **WHERE** s.shopper\_id = '10000'
23. **ORDER** **BY** so.Order\_date **DESC**
24. ;



* as with question a) the SELECT function is used, alongside PRINTF(), to print the price with the necessary ‘£’.
* FROM is used to determine where the data should come from.
* The order information is consolidated from the relevant tables using inner joins, using the abbreviations previously designated in the SELECT function.
* WHERE is then used to filter the results as per the brief
* Then ORDER BY is used to arrange the results in descending order, by the date they were produced.

c)

1. **SELECT**
2. s.Seller\_account\_ref **AS** [seller Account Ref]
3. ,s.Seller\_name  **AS** [Seller **Name**]
4. ,p.product\_code **AS** [Product Code]
5. ,p.product\_description **AS** [Product Description]
6. ,PRINTF("£%.2f",(op.quantity\*op.price)) **AS** [total Sales]
7. ,op.quantity
8. **FROM** sellers s
9. **INNER** JOIN
10. ordered\_products op **ON** op.seller\_id = s.seller\_id
11. **INNER** JOIN
12. products p **ON** p.product\_id = op.product\_id
13. **INNER** JOIN
14. shopper\_orders so **ON** so.order\_id = op.order\_id
15. **WHERE** so.order\_date > '2019-05-01' OR op.quantity = NULL
16. ;



d)

1. **SELECT**
2. category\_description **AS** [category description]
3. ,product\_description **AS** [product description]
4. ,product\_code **AS** [product code]
5. ,IFNULL(AVG(op.quantity),'0' ) **AS** [average quantity sold]
6. **FROM** Products p
7. **INNER** JOIN
8. categories c **ON** c.category\_id = p.category\_id
9. **INNER** JOIN
10. ordered\_products op **ON** op.product\_id = p.product\_id
11. **GROUP** **BY** category\_description
12. **HAVING** AVG(op.quantity) > op.quantity
13. ;

