

## EBUS3030 Assignment 2

Stavros Karmaniolos c3160280@uon.edu.au

Jay Rovacsek c3146220@uon.edu.au

Jacob Litherland c3263482@uon.edu.au

Edward Lonsdale c3252144@uon.edu.au

October 24, 2018

# Contents

<b>1</b>	<b>Assignment Overview &amp; Requirements</b>	<b>2</b>
<b>2</b>	<b>Executive Summary</b>	<b>4</b>
2.1	Datamart Business Rules . . . . .	5
<b>3</b>	<b>Data Model</b>	<b>6</b>
3.0.1	Database Schema . . . . .	6
3.0.2	EER Diagram . . . . .	7
<b>4</b>	<b>Data Load Process (ETL/ELT)</b>	<b>8</b>
4.1	Quality Assurance Processes . . . . .	9
4.2	Assumptions and Reasoning . . . . .	10
4.2.1	Item . . . . .	10
4.2.2	ReceiptItem . . . . .	10
4.2.3	Receipt . . . . .	10
4.2.4	Staff . . . . .	10
4.2.5	Customer . . . . .	10
4.2.6	Office . . . . .	10
<b>5</b>	<b>Base Analysis</b>	<b>11</b>
5.1	Notes on Analysis . . . . .	11
5.2	Raw Results . . . . .	11
5.2.1	Total Number of Sales . . . . .	11
5.2.2	Total Items Sold . . . . .	11
5.2.3	Discounted Sales Ratio . . . . .	12
5.2.4	Total Sales Value per Staff Member . . . . .	13
5.2.5	Average Value Per Sale . . . . .	13
5.3	Best location based on Items sold and total revenue . . . . .	14
5.4	Total Number of Customers . . . . .	15
5.5	Top Team-member(s) Analysis . . . . .	15
5.6	Items Per Sale . . . . .	15
5.7	Item Popularity . . . . .	15
5.8	Worst Performing Item . . . . .	15
<b>6</b>	<b>Conclusion and Recommendations</b>	<b>16</b>
	<b>References</b>	<b>17</b>
<b>7</b>	<b>Appendix</b>	<b>18</b>
7.1	C# Code Excerpt: Cleaning . . . . .	18
7.2	C# Code Excerpt: SQL Generation . . . . .	21

# 1 Assignment Overview & Requirements

EBUS3030 – Assignment 2

## Business Intelligence - EBUS3030 Assignment 2

### Assignment Outcomes

This assignment requires multiple outputs to be created to exhibit your understanding of business intelligence/data analysis through an example ‘real world’ question that is comparable to what you may be asked of you as you become an IT professional.

Key outcomes to be delivered are: Data Modelling/ETL to get the data in a usable format, Output of your analysis, Report summarising your findings and a presentation to the class of your work. The presentation is expected to concentrate more on your findings/recommendations as if it were a situation where you are presenting the response to the CEO.

### Assignment Question

The CEO of ‘BIA Inc’ had been speaking to the Sales Executive and had heard about some recent work you had completed and thought you might be able to assist them with a problem they have.

*“I’ve heard that you helped the Sales Exec recently with understanding more about our Newcastle site. I would now like your help to get a handle on the whole business. As you are aware, the Newcastle office is just 1 of 10 sites we have across the country. Unfortunately, sales in some items have dropped across the country in recent years and we are currently running at a loss.*

*We need to consider consolidating our company offices. We need to reduce costs for the longevity of the company as a whole. I need you to get some numbers together around the performance of our 10 offices, so that I can factor this information into any decision regarding which office (or offices) we might consider closing.*

*I would like a summary of recent numbers and some trend analysis as well please. It would be great if you could also project sales for the next 12 months for each office as well. It would be helpful if you could indicate the 3 most popular and 3 least popular items in each of our stores, as well as the worst performing items for the company as a whole.*

*As you can imagine, this is a very sensitive topic so, as part of your response I want you to provide the justification as to which office we may close. Our decision will upset some people and I want to make sure we have all the background information on hand. If you can provide a Ranking of offices based on your analysis that would be wonderful.*

*.... I believe you started to bring together a data store of this information from the Newcastle Office, can you expand that and load all of the sales information for all offices and complete your analysis.*  
”

### Assignment Deliverables

Using the data file provided in Excel and notes about the data (*Assignment 2 – data.xlsx*), you are required to complete the following elements as part of the assignment.

- Data Model/Data Load Process
  - Provide an overview of the data model & ETL process completed to get the data ready for analysis
  - Ensure you record any assumptions you have made as part of this component and your reasoning behind the assumption.
- Analysis including any predictive work undertaken
  - Provide the SQL and raw output of your base analysis
  - Provide workings of the predictive work you completed for the trending & prediction on future sales.
  - Ensure you record any assumptions you have made as part of this component and your reasoning behind the assumption (this includes answers to any relevant questions put to management (your lecturer) during workshops.

#### EBUS3030 – Assignment 2

- Executive Summary & Presentation in response to business question.
  - Provide an Executive report and Dashboard
    - Your Executive report should include a short Executive brief/summary that presents a clear concise response back to the CEO question about possible downsizing of operations including evidence/justification.
    - A dashboard should allow quick comparisons between the sites to be undertaken as well as contain at least one element of 'predictive' analysis
    - Present the material as if it is to be consumed in a formal boardroom meeting
      - All members of the team need to participate in a (15 – 20 minute) presentation to be given as part of the lab in Week 11.
      - Please be aware that any of the company Executive may ask questions as you present your findings.

*NB: As part of your response, you should also specifically include any assumptions/external information you have made/used throughout the process as well as any quality processes/checking you have completed or limitations you discovered.*

**Breakup of assignment Marks (total course mark for assignment = Assignment Part B submission (28% + Presentation Two (7%) = 35%).**

Assignment Component	Percentage Allocation
Data Model/ETL	10%
Core Analysis	50%
Executive Summary/Evidence	25%
Dashboard	15%
	100%

#### **Key Documents Required & Format**

You are required to upload all files in a single zip file (including any presentation items for the team delivery within the lab) via blackboard to the Assignment Two drop folder by 12 noon, Thursday 25<sup>th</sup> October. You will also be required to submit a paper copy of your report at the beginning of the presentation workshop (make sure this is printed well before the workshop and has a group Assessment Cover sheet signed by ALL team members).

NB: Only 1 load per team only but it should contain all of the deliverable items.

Your data model should include a printout of an ER diagram using the notation described in lectures. It should also include a printout of your SQL schema showing Primary and foreign keys, as well as all attributes.

Your presentation is worth 7% of the course mark. It should simulate presenting the report to management. You should time your presentation to be between 12-15 minutes with 5-10 minutes for questions. Your presentation should include a demonstration of your dashboard, results and recommendations from your analysis. Your presentation marks will contain components for organisation, comprehension of presented results, and timing.

## 2 Executive Summary

## 2.1 Datamart Business Rules

The following business rules were provided to be used in the context of this assignment:

- At BIA all customers interacts are in an online environment. We only support electronic orders.
- Returning Customers can provide POI information via the web interface and look up their record and that will flow with the sale
- The sales associate can complete the order form/sale for the client.
- Each sale will have a receipt number/id.
- A receipt can have many line items
- Each line item can only be for a single item, but the customer can purchase multiples of the same item.
- After consultation with your team, we have made the following change to discount applied to sales: Where a customer has multiple line items, any sale with 5 or more row items (containing at least 5 different items) is provided a 5% discount.
- The system automatically handles the total for the sale by looking up the item, then multiplying the costs per item by number purchased, and then should store this final field total as a record in the system (but should also be able to see clearly sales that were provided a discount.
- Store Item prices can change at any point, however the price the customer pays is the amount listed for the store item that is sold on the sale date. We need to keep a record of all store item prices historically so that we can determine what the store item price was at any particular past date.
- Only one BIA sales assistant can be attributed to any receipt.
- Customers may visit multiple stores for purchases (ie they are not locked to a particular store). As a result, all customer records are replicated across all stores, so they do not need to be re-recorded at a store by store level.

With these considerations in mind, the following report was created to outline the discovery, creation and polish to satisfy the assignment requirements.

### 3 Data Model

The following section outlines the models used in the design and creation of the database. It includes the EER Model with relations, attributes and relationships as well as the database schema from Microsoft SQL Server Management Studio.

#### 3.0.1 Database Schema

The below data model is only a suggestion and is still subject to change into the future. A full create script can be found in the [appendix](#)



It must be noted that the structure of this data model is less than efficient, and it would be expected in a datamart situation that only at lower levels of data would this schema remain responsive in the manner it is now, as the outline suggests the datamart is not necessarily the most suitable design for future use, however suits very well currently.

It would be expected that only at extremely large data sets would this model prove a bad design. In such cases a model more representative of the snowflake or star schema would be heavily advised.

### 3.0.2 EER Diagram

An EER diagram of the suggested data model is provided below.





## 4 Data Load Process (ETL/ELT)

Initial import of the data supplied in the xlsx file generated a very basic table that allowed us to analyze the data for potential outliers, confirm the business requirements of the data and then create tables from which the data model was derived.

The Imported table structure was as follows:

Assignment2Data			
	Column Name	Data Type	Allow Nulls
	Sale_Date	datetime2(7)	<input type="checkbox"/>
	Reciept_Id	int	<input type="checkbox"/>
	Customer_ID	nvarchar(50)	<input type="checkbox"/>
	Customer_First_Name	nvarchar(50)	<input type="checkbox"/>
	Customer_Surname	nvarchar(50)	<input type="checkbox"/>
	Staff_ID	nvarchar(50)	<input type="checkbox"/>
	Staff_First_Name	nvarchar(50)	<input type="checkbox"/>
	Staff_Surname	nvarchar(50)	<input type="checkbox"/>
	Staff_office	int	<input type="checkbox"/>
	Office_Location	nvarchar(50)	<input type="checkbox"/>
	Reciept_Transaction_Row_ID	int	<input type="checkbox"/>
	Item_ID	int	<input type="checkbox"/>
	Item_Description	nvarchar(50)	<input type="checkbox"/>
	Item_Quantity	int	<input type="checkbox"/>
	Item_Price	float	<input type="checkbox"/>
	Row_Total	float	<input type="checkbox"/>
			<input type="checkbox"/>

A decision to leave this initial import table as default was made to allow easy reference to the initially supplied excel data file.

## 4.1 Quality Assurance Processes

After performing a basic query to determine if, like our previous project, multiple items of the same ID on a single receipt existed it was apparent that the same issue existed in this data set as did in the previous.

To mitigate this, the team developed a simple update to the ReceiptItem was performed, to ensure that a discount would not be applied to a receipt with less than five items, when potentially the receipt did not actually have more than four distinct items. The following query was used to determine this information:

```
1  -- Verify that no receipt has duplicate ItemIds and all are unique per order
2  SELECT *
3  FROM
4  (
5      SELECT [ReceiptItem].[ReceiptId],
6      COUNT([ReceiptItem].[ReceiptId]) AS 'ItemCount',
7      COUNT(DISTINCT [ReceiptItem].[ItemId]) AS 'ItemIdCount'
8      FROM [ReceiptItem]
9      GROUP BY [ReceiptItem].[ReceiptId]) AS SubQuery
10 WHERE [SubQuery].[ItemIdCount] != [SubQuery].[ItemCount]
```

Returning no entries once the data was cleaned with a [C# Script](#) internally developed. This took into account not only checking for data integrity but generated new sql to facilitate [item consolidation](#). These scripts have been listed in the appendices of this report.

## **4.2 Assumptions and Reasoning**

The following section outlines our assumptions and the reasoning behind certain decisions made by the team at any stage of the project. The team will always try to apply business rules supplied by the firm to any of the decisions made however this is not always possible. In this case, the reasoning has been listed in the section that follows.

### **4.2.1 Item**

There were no major assumptions made for the items. We followed the feedback received from the previous project and used the associative entity of ReceiptItem to satisfy requirements of sale price and quantity.

### **4.2.2 ReceiptItem**

As suggested above, the ReceiptItem table was created to satisfy feedback received on our previous project , allows for quantity and price association.

### **4.2.3 Receipt**

The receipt had no changes in comparison to the previous analysis, the changes occurred in related entities and did not affect the receipt table.

### **4.2.4 Staff**

Staff were left untouched, the relation to location however did play a large role in this analysis, and a major assumption of a staff member only being at one location at a time or being directly related only to the one location was made.

This assumption normally would not be made, but assurances of data cleanliness were made, and the C# script did not find any discrepancies between our assumption and the supplied information.

### **4.2.5 Customer**

We made assumptions that customers are likely to shop at only the one store. Given the geographic distances between stores generally being larger than we deemed most people would be likely to travel.

### **4.2.6 Office**

Our analysis assumed future office potential growth to be related to the locality of the store. Such an assumption may suggest that we weighed in favour of stores in high population areas if the analysis was borderline for example: Newcastle and Broken Hill; Newcastle would be favoured as the growth potential was assumed to be greater and more predictable into the future.

## 5 Base Analysis

### 5.1 Notes on Analysis

Analysis was performed using Microsoft SQL Management Studio, with a flat file of the original data being imported into the table as defined [here](#), as with the previous assignment utilisation of decimal(18.7) was our choice over using the money type, again citing the imprecision of money in MSSQL/TSQL [1]

Source files to immitate the analysis below is either included inline or in sources files in the final submission.

### 5.2 Raw Results

#### 5.2.1 Total Number of Sales

The total number of sales per staff member were considered with the following sql query:

```
1 SELECT COUNT(*) AS 'Sales Count'
2     , s.StaffId
3     , s.StaffFirstName
4     , s.StaffSurname
5     , o.OfficeId
6     , o.OfficeLocation
7 FROM Receipt r
8     INNER JOIN ReceiptItem ri
9         ON r.ReceiptId = ri.ReceiptId
10    INNER JOIN Item i
11        ON i.ItemId = ri.ItemId
12    INNER JOIN Staff s
13        ON s.StaffId = r.ReceiptStaffId
14    INNER JOIN Office o
15        ON o.OfficeId = s.StaffOfficeId
16 GROUP BY s.StaffId
17         , s.StaffFirstName
18         , s.StaffSurname
19         , o.OfficeId
20         , o.OfficeLocation
21 ORDER BY 'Sales Count' DESC;
```

Sales Count	StaffId	StaffFirstName	StaffSurname	OfficeId	OfficeLocation
645	S190	Samuel	Anderson	4	Sydney
628	S196	Devin	Brown	6	Grafton
628	S122	Austin	Morris	6	Grafton
608	S101	Jenna	Cox	5	Port Macquarie
607	S45	Emma	Gutierrez	3	Cessnock

#### 5.2.2 Total Items Sold

The total items attributed to each staff member were considered also, determined by the query:

```
1 SELECT SUM(ri.ReceiptItemQuantity) AS 'Item Count'
2     , s.StaffId
3     , s.StaffFirstName
4     , s.StaffSurname
5     , o.OfficeId
6     , o.OfficeLocation
7 FROM Receipt r
8     INNER JOIN ReceiptItem ri
9         ON r.ReceiptId = ri.ReceiptId
10    INNER JOIN Staff s
11        ON s.StaffId = r.ReceiptStaffId
```

```

12         INNER JOIN Office o
13                 ON o.OfficeId = s.StaffOfficeId
14 GROUP BY s.StaffId
15           , s.StaffFirstName
16           , s.StaffSurname
17           , o.OfficeId
18           , o.OfficeLocation
19 ORDER BY 'Item Count' DESC;

```

Item Count	StaffId	StaffFirstName	StaffSurname	OfficeId	OfficeLocation
4005	S190	Samuel	Anderson	4	Sydney
3796	S122	Austin	Morris	6	Grafton
3730	S101	Jenna	Cox	5	Port Macquarie
3716	S45	Emma	Gutierrez	3	Cessnock
3684	S108	Isaiah	Powell	9	Wagga Wagga

### 5.2.3 Discounted Sales Ratio

Consideration of the number of sales made by each staff member was also made, the following query yielding the results we required:

```

1  SELECT s.StaffId
2         , s.StaffFirstName
3         , s.StaffSurname
4         , o.OfficeId
5         , o.OfficeLocation
6         , SUM(SubQuery.[Discounted Sales]) AS 'Discounted Sales'
7         , SUM(SubQuery.[Standard Sales]) AS 'Standard Sales'
8 FROM (
9     SELECT CAST(
10         CASE
11             WHEN COUNT(ri.[ReceiptItemQuantity]) >= 5
12                 THEN 1
13             ELSE 0
14             END AS INT) AS 'Discounted Sales',
15     CAST(
16         CASE
17             WHEN COUNT(ri.[ReceiptItemQuantity]) >= 5
18                 THEN 0
19             ELSE 1
20             END AS INT) AS 'Standard Sales',
21     r.ReceiptId
22 FROM Receipt r
23         INNER JOIN ReceiptItem ri
24                 ON r.ReceiptId = ri.ReceiptId
25         INNER JOIN Item i
26                 ON i.ItemId = ri.ItemId
27     GROUP BY r.ReceiptId
28 ) AS SubQuery
29     INNER JOIN Receipt r
30             ON SubQuery.ReceiptId = r.ReceiptId
31     INNER JOIN ReceiptItem ri
32             ON r.ReceiptId = ri.ReceiptId
33     INNER JOIN Staff s
34             ON s.StaffId = r.ReceiptStaffId
35     INNER JOIN Office o
36             ON o.OfficeId = s.StaffOfficeId
37 GROUP BY s.StaffId

```

```

38         , s.StaffFirstName
39         , s.StaffSurname
40         , o.OfficeId
41         , o.OfficeLocation
42 ORDER BY [Discounted Sales];

```

StaffId	StaffFirstName	StaffSurname	OfficeId	OfficeLocation	Discounted Sales	Standard Sales
S51	Haley	Taylor	7	Dubbo	263	91
S161	Jason	Wood	7	Dubbo	273	117
S17	Daniel	Baker	1	Newcastle	278	112
S135	Lexi	James	4	Sydney	280	100
S73	John	White	2	Maitland	286	121

#### 5.2.4 Total Sales Value per Staff Member

```

1 SELECT CAST(
2     CASE
3     WHEN COUNT(ri.[ReceiptItemQuantity]) >= 5
4         THEN SUM(ri.[SalePrice] * ri.[ReceiptItemQuantity]) * 0.95
5     ELSE SUM(ri.[SalePrice] * ri.[ReceiptItemQuantity])
6     END AS decimal(19,5)) AS 'Sales Totals'
7     , s.StaffId
8     , s.StaffFirstName
9     , s.StaffSurname
10    , o.OfficeId
11    , o.OfficeLocation
12 FROM Receipt r
13     INNER JOIN ReceiptItem ri
14         ON r.ReceiptId = ri.ReceiptId
15     INNER JOIN Item i
16         ON i.ItemId = ri.ItemId
17     INNER JOIN Staff s
18         ON s.StaffId = r.ReceiptStaffId
19     INNER JOIN Customer c
20         ON c.CustomerId = r.ReceiptCustomerId
21     INNER JOIN Office o
22         ON o.OfficeId = s.StaffOfficeId
23 GROUP BY s.StaffId
24         , s.StaffFirstName
25         , s.StaffSurname
26         , o.OfficeId
27         , o.OfficeLocation
28 ORDER BY 'Sales Totals' DESC;

```

Sales Totals	StaffId	StaffFirstName	StaffSurname	OfficeId	OfficeLocation
\$77152.49250	S187	Savannah	Jones	8	Wollongong
\$75113.60250	S45	Emma	Gutierrez	3	Cessnock
\$72475.45250	S178	Kaitlyn	Nguyen	2	Maitland
\$71814.20500	S122	Austin	Morris	6	Grafton
\$71497.09500	S71	Danielle	Myers	6	Grafton

#### 5.2.5 Average Value Per Sale

```

1 SELECT (CAST(
2     CASE
3     WHEN COUNT(ri.[ReceiptItemQuantity]) >= 5

```

```

4          THEN SUM(ri.[SalePrice] * ri.[ReceiptItemQuantity]) * 0.95
5      ELSE SUM(ri.[SalePrice] * ri.[ReceiptItemQuantity])
6      END AS decimal(19,5)) / COUNT(r.ReceiptId)) AS 'Sales Average'
7      , s.StaffId
8      , s.StaffFirstName
9      , s.StaffSurname
10     , o.OfficeId
11     , o.OfficeLocation
12 FROM Receipt r
13 INNER JOIN ReceiptItem ri
14         ON r.ReceiptId = ri.ReceiptId
15 INNER JOIN Item i
16         ON i.ItemId = ri.ItemId
17 INNER JOIN Staff s
18         ON s.StaffId = r.ReceiptStaffId
19 INNER JOIN Office o
20         ON o.OfficeId = s.StaffOfficeId
21 GROUP BY s.StaffId
22         , s.StaffFirstName
23         , s.StaffSurname
24         , o.OfficeId
25         , o.OfficeLocation
26 ORDER BY 'Sales Average' DESC;

```

Sales Average	StaffId	StaffFirstName	StaffSurname	OfficeId	OfficeLocation
\$138.40	S109	Nicole	Hernandez	10	Broken Hill
\$134.88	S187	Savannah	Jones	8	Wollongong
\$134.71	S52	Isabella	Rivera	5	Port Macquarie
\$134.53	S165	Marcus	Ross	2	Maitland
\$134.12	S199	Maria	Smith	5	Port Macquarie

### 5.3 Best location based on Items sold and total revenue

```

1  -- Item Count, total revenue, average revenue per item sold By Office Location no
   discount
2  SELECT SUM(ri.ReceiptItemQuantity) AS ItemCount
3         , o.OfficeId
4         , o.OfficeLocation
5         , SUM(ri.[SalePrice] * ri.[ReceiptItemQuantity]) AS Revenue
6         , Cast (SUM(ri.[SalePrice] * ri.[ReceiptItemQuantity]) as decimal)/SUM(ri
   .ReceiptItemQuantity) as AverageRevenue
7  FROM Receipt r
8     INNER JOIN ReceiptItem ri
9             ON r.ReceiptId = ri.ReceiptId
10    INNER JOIN Staff s
11            ON s.StaffId = r.ReceiptStaffId
12    INNER JOIN Office o
13            ON s.StaffOfficeId = o.OfficeId
14  GROUP BY o.OfficeId
15         , o.OfficeLocation
16  ORDER BY ItemCount DESC;

```

ItemCount	OfficeId	OfficeLocation	Revenue
97020	9	Wagga Wagga	\$1902270.85
65637	2	Maitland	\$1309222.55
65568	10	Broken Hill	\$1270277.65
59946	1	Newcastle	\$1167716.60
59229	4	Sydney	\$1171929.90
56429	5	Port Macquarie	\$1150447.25
52315	3	Cessnock	\$1059141.65
51533	6	Grafton	\$1054211.90
49456	7	Dubbo	\$983439.45
46433	8	Wollongong	\$938812.30

## 5.4 Total Number of Customers

---

Sales Count	StaffId	StaffFirstName	StaffSurname
-------------	---------	----------------	--------------

## 5.5 Top Team-member(s) Analysis

Analysis over all stores, correlation to store we want to nuke?

---

Sales Count	StaffId	StaffFirstName	StaffSurname
-------------	---------	----------------	--------------

## 5.6 Items Per Sale

---

Sales Count	StaffId	StaffFirstName	StaffSurname
-------------	---------	----------------	--------------

## 5.7 Item Popularity

Top 3 best and worst items overall, correlation to any stores?

---

Sales Count	StaffId	StaffFirstName	StaffSurname
-------------	---------	----------------	--------------

## 5.8 Worst Performing Item

Correlation to store?

---

Sales Count	StaffId	StaffFirstName	StaffSurname
-------------	---------	----------------	--------------



## 6 Conclusion and Recommendations

## References

- [1] Reasons against TSQL Money type: Stackoverflow User; *SQLMenace* <https://stackoverflow.com/questions/582797/should-you-choose-the-money-or-decimalx-y-datatypes-in-sql-server>
- [2] Microsoft TSQL documentation of Decimal/Numeric types <https://docs.microsoft.com/en-us/sql/t-sql/data-types/decimal-and-numeric-transact-sql?view=sql-server-2017>
- [3] Microsoft documentation: WITH common\_table\_expression (Transact-SQL) <https://docs.microsoft.com/en-us/sql/t-sql/queries/with-common-table-expression-transact-sql?view=sql-server-2017>
- [4] Upselling - Business Dictionary <http://www.businessdictionary.com/definition/upselling.html>

## 7 Appendix

### 7.1 C# Code Excerpt: Cleaning

```
1 public Tuple<Dictionary<int, Receipt>, Dictionary<int, Receipt>, int> GenerateReceipts(DataSet dataSet)
2 {
3     var receipts = new Dictionary<int, Receipt>();
4     var invalidReceipts = new Dictionary<int, Receipt>();
5     var counter = 0;
6
7     // Iterate over each sheet in the dataset, in our case it's only one anyway.
8     foreach (DataTable table in dataSet.Tables)
9         // Iterate over each row in the dataset, it seems this iterates until it finds row with no values in any
            column.
10        foreach (DataRow row in table.Rows)
11        {
12            // Ignore the header of the table
13            if (counter > 0)
14            {
15                // Generation of receipt from dataset, ignoring transaction row id as it is not
16                // relevant and we are consolidating the duplicate items into total quantities anyway.
17                var receipt = new Receipt
18                {
19                    Id = Convert.ToInt32(row[1]),
20                    Customer = new Customer
21                    {
22                        FirstName = row[3].ToString(),
23                        Id = row[2].ToString(),
24                        Surname = row[4].ToString()
25                    },
26                    Staff = new Staff
27                    {
28                        OfficeId = Convert.ToInt32(row[8]),
29                        Id = row[5].ToString(),
30                        FirstName = row[6].ToString(),
31                        Surname = row[7].ToString()
32                    },
33                    SaleDate = (DateTime)row[0],
34                    Items = new Dictionary<int, Item>
35                    {
36                        {
37                            Convert.ToInt32(row[11]), new Item
38                            {
39                                Id = Convert.ToInt32(row[11]),
40                                Description = row[12].ToString(),
41                                Price = (double) row[14],
42                                Quantity = Convert.ToInt32(row[13])
43                            }
44                        }
45                    },
46                    Office = new Office
47                    {
48                        Id = Convert.ToInt32(row[8]),
49                        OfficeLocation = ParseEnum<Location>(row[9].ToString())
50                    }
51                };
52            }
53        }
54    }
55 }
56
```

```

57 // Check if receipts either is empty or doesn't have a receipt
58 // with the same receipt id already, if so add the receipt to
59 // receipts .
60 if (receipts.Count == 0 || !receipts.ContainsKey(receipt.Id))
61 {
62     AddToReceipts(receipts, receipt);
63     Console.WriteLine($"@Added new receipt: {receipt.Id}");
64 }
65 //
66 else if (receipts.ContainsKey(receipt.Id))
67 {
68     var existingReceipt = receipts[receipt.Id];
69
70     // Be warned; below is a fucking mess.
71
72     // Check all staff properties match the existing receipt properties.
73     if (existingReceipt.Staff.Id != receipt.Staff.Id
74         || existingReceipt.Staff.FirstName != receipt.Staff.FirstName
75         || existingReceipt.Staff.Surname != receipt.Staff.Surname)
76     {
77         AddToReceipts(invalidReceipts, receipt);
78         Console.WriteLine($"Staff Id mismatch on receipt: {existingReceipt.Id}, " +
79             $"Staff Ids: {existingReceipt.Staff.Id} and {receipt.Staff.Id}, " +
80             $"Staff Names: {existingReceipt.Staff.FirstName} and
81                 {receipt.Staff.FirstName}, " +
82             $"Staff Surnames: {existingReceipt.Staff.Surname} and
83                 {receipt.Staff.Surname}, ");
84         continue;
85     }
86
87     // Check all office properties match the existing receipt properties.
88     if (existingReceipt.Office.Id != receipt.Office.Id
89         || existingReceipt.Office.OfficeLocation != receipt.Office.OfficeLocation)
90     {
91         AddToReceipts(invalidReceipts, receipt);
92         Console.WriteLine($"Office Id mismatch on receipt: {existingReceipt.Id}, " +
93             $"Office Ids: {existingReceipt.Office.Id} and {receipt.Office.Id}, " +
94             $"Office Locations: {existingReceipt.Office.OfficeLocation} and
95                 {receipt.Office.OfficeLocation}");
96         continue;
97     }
98
99     // Check all date properties match the existing receipt properties.
100    if (existingReceipt.SaleDate != receipt.SaleDate)
101    {
102        AddToReceipts(invalidReceipts, receipt);
103        Console.WriteLine($"Sale date mismatch on receipt: {existingReceipt.Id}, " +
104            $"Dates: {existingReceipt.SaleDate} and {receipt.SaleDate}");
105        continue;
106    }
107
108
109
110
111
112
113
114

```

```

115 // Check all customer properties match the existing receipt properties.
116 if (existingReceipt.Customer.Id != receipt.Customer.Id
117     || existingReceipt.Customer.FirstName != receipt.Customer.FirstName
118     || existingReceipt.Customer.Surname != receipt.Customer.Surname)
119 {
120     AddToReceipts(invalidReceipts, receipt);
121     Console.WriteLine($"Customer mismatch on receipt: {existingReceipt.Id}, " +
122         $"Customers Ids: {existingReceipt.Customer.Id} and {receipt.Customer.Id}, "
123         +
124         $"Customers Names: {existingReceipt.Customer.FirstName} and
125         {receipt.Customer.FirstName}, " +
126         $"Customers Surnames: {existingReceipt.Customer.Surname} and
127         {receipt.Customer.Surname}, ");
128     continue;
129 }
130 // If not item mismatches exist, we need to determine if the item
131 // exists in both the compared receipts.
132 var key = receipt.Items.First().Key;
133 var value = receipt.Items.First().Value;
134 // Check if the item exists in the existing receipt already.
135 if (existingReceipt.Items.ContainsKey(receipt.Items.First().Key))
136 {
137     var quantity = receipt.Items.First().Value.Quantity;
138     Console.WriteLine($"Item exists in current receipt with Id: {key}, " +
139         $"updated item quantity from: {existingReceipt.Items[key].Quantity} " +
140         $"to new quantity: {existingReceipt.Items[key].Quantity + quantity}");
141     // Update quantity on item in existing receipt.
142     existingReceipt.Items[key].Quantity += quantity;
143     receipts[existingReceipt.Id] = existingReceipt;
144 }
145 else
146 {
147     // Update the receipt with the new item added.
148     existingReceipt.Items.Add(key, value);
149     Console.WriteLine($"@ Added item with Id: {value.Id} to existing receipt: {receipt.Id}");
150 }
151 }
152 }
153 counter++;
154 }
155
156 return new Tuple<Dictionary<int, Receipt>, Dictionary<int, Receipt>, int>(receipts, invalidReceipts,
157     counter);
158 }

```

## 7.2 C# Code Excerpt: SQL Generation

```
1 public string GenerateSQL(Dictionary<int, Receipt> receipts)
2 {
3     var output = new StringBuilder();
4
5     foreach(var receipt in receipts)
6     {
7         var itemTotalPrices = new Dictionary<int, double>();
8
9         foreach (var item in receipt.Value.Items)
10        {
11            var totalItemPrice = item.Value.Price * item.Value.Quantity;
12            itemTotalPrices.Add(item.Key, totalItemPrice);
13        };
14
15        var total = itemTotalPrices.Sum(x => x.Value);
16        var discountTotal = (itemTotalPrices.Count >= 5) ? total * 0.95 : 0;
17
18        var sql = new StringBuilder(@"
19
20
21        INSERT INTO [Receipt]
22        VALUES('{'receipt.Value.SaleDate.ToString("G",CultureInfo.CreateSpecificCulture("en-us"))}',
23                {'receipt.Key},
24                {'receipt.Value.Customer.Id}',
25                {'receipt.Value.Staff.Id'},
26                {total},
27                {discountTotal});");
28
29        foreach (var item in itemTotalPrices)
30        {
31            sql.Append(@"
32
33        INSERT INTO [ReceiptItem]
34        VALUES( {'receipt.Key},
35                {item.Key},
36                {receipt.Value.Items.Select(x => x.Value).Where(y => y.Id == item.Key).FirstOrDefault().Quantity},
37                {receipt.Value.Items.Select(x => x.Value).Where(y => y.Id == item.Key).FirstOrDefault().Price});");
38        }
39
40        output.Append(sql);
41    };
42
43    return output.ToString();
44 }
```