Fwd_ JPRP-19-07-24_30 _ £Vikas Andotra_GD604_data Collecton and analaysis.docx

by Turnitin LLC

Submission date: 21-Jul-2024 12:01AM (UTC+0530)
Submission ID: 2416827020
File name: 2024_07_20_Fwd__JPRP-19-07-24_30___Vikas__e48ad575e03e768d.docx (999.87K)
Word count: 3429
Character count: 18423

ASSESSMENT 2

Table of Contents

Introduction 4

Task A –Data Transformation 4

Task A(a) - Load the dataset into a DataFrame 4

Task A(b) - Show the first few rows of the loaded dataset 5

Task A(c) - Apply three operations to handle missing values in the dataset 6

Task A(d) - Choose a column and perform the sorting technique 7

Task A(e) - Define a condition to filter transactions from the dataset 8

Task A(f) - Create a new column to derive additional information 9

Task A(g) - Choose the categorical column and aggregate data based on it 10

Task B – Data Analysis 11

Task B(a) - Group the dataset based on a categorical variable and calculate summary statistics 11

Task B(b) - Investigate the correlations between different variables in the dataset 12

Task B(c) - Export a dataset to a CSV file using Python or any other similar programming tool 13

Task B(d) - Perform data analysis and visualization in Excel, Python or any other similar programming tool to derive insights 14

Task B(e) - Apply inferential statistical methods to quantify the relationships between variables 16

Task C – Data Findings and Decision Support 17

Task C(a) - Analyze the results obtained from data analysis, including grouping, summarizing, investigating correlations, and applying inferential statistical methods 17

Task C(b) - Interpret the relationships between variables, summarize key findings, and identify significant trends or patterns 18

Task C(c) - Provide specific suggestions for addressing business challenges or opportunities identified in the dataset 18

Conclusion 19

References.. 20

Introduction

Data Collection and Analysis is a core course widely aimed at the effective utilization of data in management of organizations. This course also introduces different approaches of how data may be gathered from different sources and also it stresses on the ways of converting, scrutinizing and interpreting the collected data. This way, the course demonstrates that data analysis can help one identify the areas of potential development and the problems that should be solved. Such an orientation guarantees that the business can harness the power of data and implement the right tools to enhance the organizational processes. Finally, the course highlights the importance of data supporting the strategic business goals and objectives.

Task A –Data Transformation

Task A(a) - Load the dataset into a DataFrame

Task A(a) - Load the dataset into a DataFrame.

```
import pandas as pd

# Load the dataset
df = pd.read_csv('supply_chain_data.csv')
```

Figure 1: Loading the dataset into a DataFrame

(Source: Created by the learner)

The above figure displays the code to load the given data set into a DataFrame, which is the initial and common step, when using the Python data manipulation tool – Pandas. The type of data structures named DataFrames are represented in the form of labeled and rectilinear two-dimensional tables of data analogous to the concept of a spreadsheet. The first line of the code defines a method in which the Pandas library is imported as pd, which is the most often used abbreviation for it by data analysts. Pandas is specifically developed for the use of data retrieval and analysis with excellent speed (Braun and Clarke, 2021). The next one is to build a DataFrame object with the name of 'df' using the function pd.read_csv'supply_chain_data. csv'. This function fetches the data from the CSV file with the name supply_chain_data. csv and puts the data into a variable called DataFrame. This approach is famous because simple and efficient in handling of data for further analysis.

Task A(b) - Show the first few rows of the loaded dataset

Task A(b) - Show the first few rows of the loaded dataset.

```
[ ] # Show the first few rows
    print(df.head())
```

```
  Product type   SKU      Price  Availability  Number of products sold  \
0     haircare  SKU0  69.808006            55                      802   
1     skincare  SKU1  14.843523            95                      736   
2     haircare  SKU2  11.319683            34                        8   
3     skincare  SKU3  61.163343            68                       83   
4     skincare  SKU4   4.805496            26                      871   

   Revenue generated Customer demographics  Stock levels  Lead times  \
0        8661.996792            Non-binary            58           7   
1        7460.900065                Female            53          30   
2        9577.749626               Unknown             1          10   
3        7766.836426            Non-binary            23          13   
4        2686.505152            Non-binary             5           3   

   Order quantities  ...  Location Lead time  Production volumes  \
0                96  ...    Mumbai        29                 215   
1                37  ...    Mumbai        23                 517   
2                88  ...    Mumbai        12                 971   
3                59  ...   Kolkata        24                 937   
4                56  ...     Delhi         5                 414   

  Manufacturing lead time Manufacturing costs  Inspection results  \
0                      29           46.279879             Pending   
1                      30           33.616769             Pending   
2                      27           30.688019             Pending   
3                      18           35.624741                Fail   
4                       3           92.065161                Fail   

   Defect rates  Transportation modes   Routes       Costs
```

Figure 2: Showing the first few rows of the loaded dataset

(Source: Created by the learner)

The image below represents the output of the script written in the python language for Task A(b) containing the first lines of the data loaded. This output, however, involves the use of tables that are like simple spreadsheets with the labels on the columns and each row as a single record of data. The first column gives numerical tags and the second one, the 'Product type', contains names like 'haircare' and 'skincare', which imply the targeted sale of products. The subsequent columns with labels "SKUO" and "SKU" include product codes (Wang *et al*. 2020). The column labelled as "Price" possibly contains the information about product prices while the column "Avail." may refer to the stocks of the products. The last column of the table namely "Number of products sold" shows the number of products that had been sold out.

Task A(c) - Apply three operations to handle missing values in the dataset

Task A(c) - Apply three operations to handle missing values in the dataset.

```
# Identify numeric and non-numeric columns
numeric_cols = df.select_dtypes(include='number').columns
non_numeric_cols = df.select_dtypes(exclude='number').columns

# Operation 1: Fill missing values with a specific value (e.g., 0) for numeric columns
df[numeric_cols] = df[numeric_cols].fillna(0)

# Operation 2: Drop rows with missing values (Note: This will drop rows where any column has missing values)
df.dropna(inplace=True)

# Operation 3: Fill missing values with the mean of the column for numeric columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].mean())

# Verify missing values handled
print(df.isnull().sum())
```

```
Product type             0
SKU                      0
Price                    0
Availability             0
Number of products sold  0
Revenue generated        0
Customer demographics    0
Stock levels             0
Lead times               0
Order quantities         0
Shipping times           0
Shipping carriers        0
```

Figure 3: Applying three operations to handle missing values in the dataset

(Source: Created by the learner)

The associated picture shows implementation of code to handle with missing values in a dataset which is imported into the Pandas DataFrame. Another kind of data quality issue is missing values often denoted by NaN, or Not a Number. Three operations can be noticed dealing with the issue of missing values in the code. Firstly, it identifies numeric and non-numeric columns, creating two variables: two data frames: numeric_cols for column with numerical data and non_numeric_cols for the column with other types of data. By using the select_dtypes function. Second, it imputes missing values in the selected numeric columns with a fixed value, for example, 0 using the fillna function on df[numeric_cols] (Lochmiller, 2021). Thirdly, it removes any row that contains any missing value using the dropna function on DataFrame df and with inplace=True argument so that it is modified directly.

Task A(d) - Choose a column and perform the sorting technique

Task A(d) - Choose a column and perform the sorting technique.

```
[ ] # Sort the dataframe by the 'Price' column
    df_sorted = df.sort_values(by='Price')
    print(df_sorted.head())
```

```
   Product type   SKU     Price  Availability  Number of products sold  \
5      haircare  SKU5  1.699976            87                      147
28    cosmetics SKU28  2.397275            12                      394
94    cosmetics SKU94  3.037689            97                      987
74     haircare SKU74  3.170011            64                      904
97     haircare SKU97  3.526111            56                       62

    Revenue generated Customer demographics  Stock levels  Lead times  \
5         2828.348746            Non-binary            90          27
28        6117.324615                Female            48          15
94        7888.356547               Unknown            77          26
74        5709.945296                Female            41           6
97        4370.916580                  Male            46          19

    Order quantities  ...  Lead time Production volumes  \
5                 66  ...         10                104
28                24  ...         13                171
94                72  ...         12                908
74                 1  ...          1                919
97                 4  ...         10                535

    Manufacturing lead time Manufacturing costs Inspection results  \
5                        17           56.766476               Fail
28                        7           59.429382               Fail
94                       14           60.387379               Pass
74                        9           80.580852               Fail
97                       13           65.765156               Fail
```

Figure 4: Choosing a column and perform the sorting technique

(Source: Created by the learner)

The above figure represents the method for sorting a given DataFrame according to specific column in the framework of Pandas' library. In this particular case, the sorting parameters describing the nature of the sorting operation are the 'Price' column of the DataFrame. The primary operation is carried out by the line df_sorted = df. ranks, which creates a new DataFrame, df_sorted, holding the record sorted by its value in 'Price'. By default, the . sort_values() function sorts the data in the ascending order which means from the lowest to the highest price. The code also contains the line print(df_sorted. head()), which shows the beginning of the DataFrame containing sorted data identified as df_sorted. Here, head() method is used where it is constrained, usually to five rows (Peck and Olsen, 2020). This method can be useful for getting a quick look at the sorted DataFrame so that the users can be sure that sorting operation has taken place successfully.

Task A(e) - Define a condition to filter transactions from the dataset

Task A(e) - Define a condition to filter transactions from the dataset.

```python
# Filter transactions where the 'Number of products sold' is greater than 500
filtered_df = df[df['Number of products sold'] > 500]
print(filtered_df.head())
```

```
   Product type   SKU      Price  Availability  Number of products sold  \
0      haircare  SKU0  69.808006            55                      802
1      skincare  SKU1  14.843523            95                      736
4      skincare  SKU4   4.805496            26                      871
9      skincare  SKU9  64.015733            35                      980
10     skincare SKU10  15.707796            11                      996

    Revenue generated Customer demographics  Stock levels  Lead times  \
0         8661.996792            Non-binary            58           7
1         7460.900065                Female            53          30
4         2686.505152            Non-binary             5           3
9         4971.145988               Unknown            14          27
10        2330.965802            Non-binary            51          13

    Order quantities  ...  Lead time Production volumes  \
0                 96  ...         29                215
1                 37  ...         23                517
4                 56  ...          5                414
9                 83  ...         29                963
10                80  ...         18                830

    Manufacturing lead time Manufacturing costs Inspection results  \
0                        29           46.279879            Pending
1                        30           33.616769            Pending
4                         3           92.065161               Fail
9                        23           47.957602            Pending
10                        5           96.527353               Pass
```

Figure 5: Defining a condition to filter transactions from the dataset

(Source: Created by the learner)

The above figure shows that from the transactions of a dataset, it is possible to filter them depending on the number of products sold. The condition specified in the chapter targets those transactions where the 'Number of products sold' is greater than a given value. Namely, this value is assumed to be 500 with the help of the expression df['Number of products sold'] > 500. This expression will make a selection of rows where the number of products sold is more than 500. The first rows of this filtered DataFrame are printed using the print(filtered_df. head()) line, which will show transactions larger than 500 products sold (Putri and Simanjuntak, 2022). This process helps to exclude and show high-velocity transactions, which contributes to the further analysis of such data.

Task A(f) - Create a new column to derive additional information

Task A(f) - Create a new column to derive additional information.

```
# Create a new column 'Revenue per Product' derived from 'Revenue generated' / 'Number of products sold'
df['Revenue per Product'] = df['Revenue generated'] / df['Number of products sold']
print(df.head())
```

```
  Product type   SKU      Price  Availability  Number of products sold  \
0     haircare  SKU0  69.808006            55                      802
1     skincare  SKU1  14.843523            95                      736
2     haircare  SKU2  11.319683            34                        8
3     skincare  SKU3  61.163343            68                       83
4     skincare  SKU4   4.805496            26                      871

   Revenue generated Customer demographics  Stock levels  Lead times  \
0        8661.996792            Non-binary            58           7
1        7460.900065                Female            53          30
2        9577.749626               Unknown             1          10
3        7766.836426            Non-binary            23          13
4        2686.505152            Non-binary             5           3

   Order quantities  ...  Lead time Production volumes  \
0                96  ...         29                215
1                37  ...         23                517
2                88  ...         12                971
3                59  ...         24                937
4                56  ...          5                414

   Manufacturing lead time Manufacturing costs Inspection results  \
0                       29           46.279879            Pending
1                       30           33.616769            Pending
2                       27           30.688019            Pending
3                       18           35.624741               Fail
4                        3           92.065161               Fail
```

Figure 6: Creating a new column to derive additional information

(Source: Created by the learner)

The above figure displays a new column in a Pandas DataFrame for the derived details, namely, revenue per product A line of code: df[‘Revenue per Product] . The right side of the assignment outlines how the values for this new column will be determined by using the formula ‘Revenue generated/Number of products sold.’ This operation essentially entails the creation of the new column by using the results of the calculation and placing it in ‘Revenue per Product’. Last of all, the code has the statement print(df.head()), which prints the first rows of the DataFrame ‘df’ after applying the new column. This preview also gives the viewer an opportunity to see how the ‘‘Revenue per Product’’ values are being generated within the DataFrame.

Task A(g) - Choose the categorical column and aggregate data based on it

Task A(g) - Choose the categorical column and aggregate data based on it.

```
[ ] # Aggregate data based on the 'Product type' column
    aggregated_df = df.groupby('Product type').agg({
        'Revenue generated': 'sum',
        'Number of products sold': 'sum'
    }).reset_index()
    print(aggregated_df)
```

```
  Product type  Revenue generated  Number of products sold
0    cosmetics      161521.265999                    11757
1     haircare      174455.390605                    13611
2     skincare      241628.162133                    20731
```

Figure 7: Choosing the categorical column and aggregate data based on it

(Source: Created by the learner)

The above figure displays how to take data to draw values in a specifically in a Pandas DataFrame by a categorical column. This begins by choosing the ‘Product type’ column as the one to be used for grouping as well as for aggregation. Next , the DataFrame df is then grouped by this column which means resulting data are distributed according to categories in the ‘Product type’ (for example, ‘haircare,’ ‘skincare’). Which performs a number of aggregation functions on each of the groups. Notably, to arrive at the total revenue (Lemon and Hayes, 2020). It totals up the ‘Revenue generated’ column while totaling up the ‘Number of products sold’ column for each type of the product. The data in form of a new DataFrame is stored in a new variable termed as aggregated_df after being aggregated. Also to help when referring to the index of aggregated_df it is set back to 0.

Task B – Data Analysis

Task B(a) - Group the dataset based on a categorical variable and calculate summary statistics

Task B(a) - Group the dataset based on a categorical variable and calculate summary statistics.

```
# Group by 'Product type' and calculate summary statistics
grouped_stats = df.groupby('Product type').describe()
print(grouped_stats)
```

```
              Price                                                         \
              count       mean        std       min        25%        50%
Product type
cosmetics      26.0  57.361058  30.423912  2.397275  44.147347  64.311917
haircare       34.0  46.014279  28.850845  1.699976  26.796123  48.588874
skincare       40.0  47.259329  33.337844  4.078333  17.033205  37.228419

                                  Availability             ...       Costs  \
                    75%        max        count       mean ...         75%
Product type                                               ...
cosmetics     80.821379  97.760086         26.0  51.230769 ...  741.330529
haircare      69.678712  97.446947         34.0  43.264706 ...  748.054609
skincare      82.344926  99.171329         40.0  50.925000 ...  775.363776

                         Revenue per Product                                  \
                     max               count       mean         std       min
Product type
cosmetics     996.778315                26.0  31.461863   66.673142  2.112671
haircare      997.413450                34.0  73.334005  208.639573  1.366235
skincare      995.929461                40.0  20.606668   24.668552  2.255748


                    25%        50%        75%          max
Product type
cosmetics      8.084289  14.095738  22.265678   347.384522
haircare       6.957490  12.820516  42.433277  1197.218703
```

Figure 8: Grouping the dataset based on a categorical variable and calculate summary statistics

(Source: Created by the learner)

The above figure displays how a dataset can be categorized according to one variable and some statistics can be generated. In more detail, it uses the method groupby() where the column by which the data is divided is specified – df.groupby('Product type'). descriptive analysis with functions like describe() create a table of descriptive statistics of every group. These consist of count which depicts number of items in each category, mean which gives the overall mean for numerical columns within each group and std which depicts the spread of data from the mean (Braun and Clarke, 2022). Further, the table offers information on the min, 25%, 50%, 75%, and the max of the numerical columns after grouping the data. These summary measures of central tendency and dispersion of the data are essential in analysing different types of products and their distribution.

Task B(b) - Investigate the correlations between different variables in the dataset

Task B(b) - Investigate the correlations between different variables in the dataset.

```
[ ] from sklearn.preprocessing import LabelEncoder

    le = LabelEncoder()
    df_encoded = df.copy()
    for column in df_encoded.select_dtypes(include=['object']).columns:
        df_encoded[column] = le.fit_transform(df_encoded[column])

    correlation_matrix = df_encoded.corr()
    print(correlation_matrix)
```

```
Costs                           0.012572      0.032072
Revenue per Product             0.041946      0.139478

                         Transportation modes    Routes     Costs  \
Product type                        -0.073864  0.003619  0.070671
SKU                                  0.034294 -0.066807  0.138706
Price                                0.008989  0.149359  0.088501
Availability                         0.030393  0.017730 -0.027315
Number of products sold              0.075610 -0.053316 -0.036951
Revenue generated                   -0.052785 -0.002071  0.027252
Customer demographics               -0.042649  0.088044 -0.056375
Stock levels                        -0.048568  0.006626 -0.012088
Lead times                          -0.169066  0.093482  0.243686
Order quantities                    -0.025173  0.148212  0.167306
Shipping times                       0.117325 -0.105624 -0.045541
Shipping carriers                    0.020652  0.038885  0.093586
Shipping costs                      -0.116143  0.071578  0.051671
Supplier name                        0.170449 -0.135151 -0.054107
Location                             0.001275  0.055300 -0.253995
```

Figure 9: Investigating the correlations between different variables in the dataset

(Source: Created by the learner)

The above figure deals with the exploration of other measures of correlation of variables in a data set read in using Pandas DataFrame. It then imports the LabelEncoder class from the scikit-learn module of Python which assists to deal with categorical data before correlation calculations. The code then creates a new DataFrame, df_encoded, which is meant to be an encoded form of the DataFrame df so that it can accommodate categorical variables that may affect correlation calculations (Shrestha, 2021). After that, it computes the correlation matrix using df_encoded variable as the input of the function. corr(), the function for calculating correlation coefficients of all numeric columns in a DataFrame. This matrix shows the direction and the extent of the linear association between variables where the acceptable score is the range of -1 to 1.

Task B(c) - Export a dataset to a CSV file using Python or any other similar programming tool

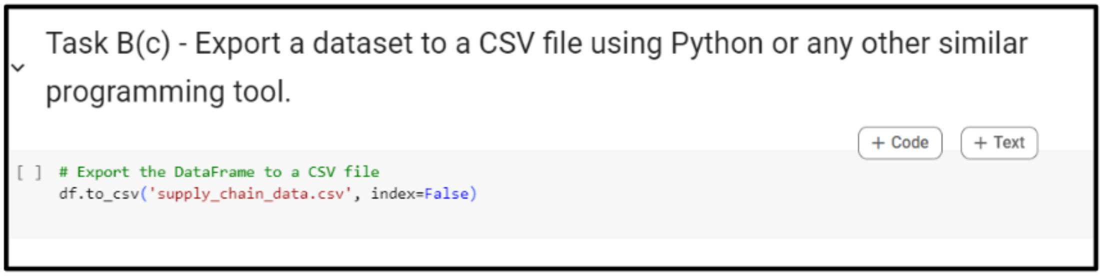

Figure 10: Exporting a dataset to a CSV file using Python or any other similar programming tool

(Source: Created by the learner)

The above figure code can be used to export a given dataset to CSV file format by using the Pandas programming library of python. The given code has the line df. to_csv('supply_chain_data. csv', index = False), where the information in the DataFrame df is to be exported into a CSV file with the name of the file being supply_chain_data. csv. This single line of code makes use of the attributes of the DataFrame and symmetrically writes the data in the format of Comma Separated Value to its object (Adeoye-Olatunde and Olenik, 2021). The parameter index=False makes it possible to exclude the annoyance of row indices that usually accompany DataFrames. This option is very useful when the row indices are not to be used in the final output and we end up with cleaner csv file.

Task B(d) - Perform data analysis and visualization in Excel, Python or any other similar programming tool to derive insights

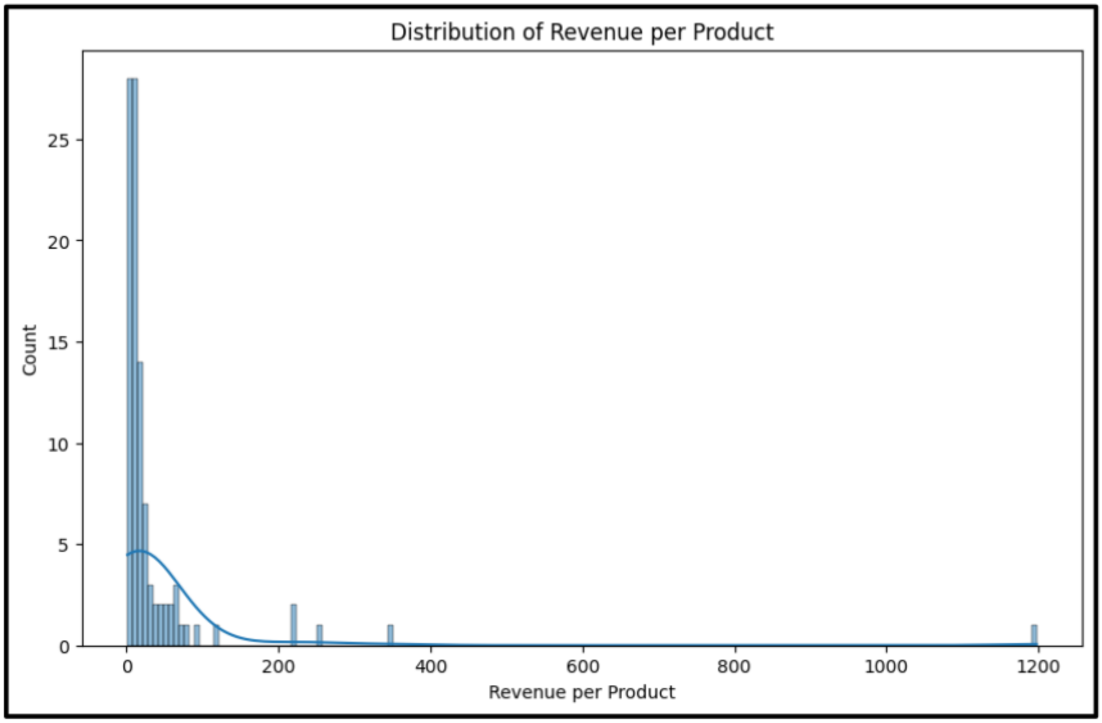

Figure 11: Visualization 1: Distribution of 'Revenue per Product'

(Source: Created by the learner)

The above figure related to two numerical variables. Where the dots are distributed in the scatter plot is the thoughts or ideas about each product. The horizontal axis holds unknown parameters, which may be selected as product code, price, or quantity sold, and the vertical axis illustrates the income per unit of the product. The x-axis shows that the range of revenues was high; hence, the variability of the products to generate revenues was also high. Although the nature of the horizontal axis is unclear, the plot shows that some products have much higher revenues per unit than others. This spread indicates a concept that there exists variation in the performance of the products. While there is some ambiguous information related to the exact nature of these axes it is clear that the spread represents a rather diverse set of revenue values in the dataset.

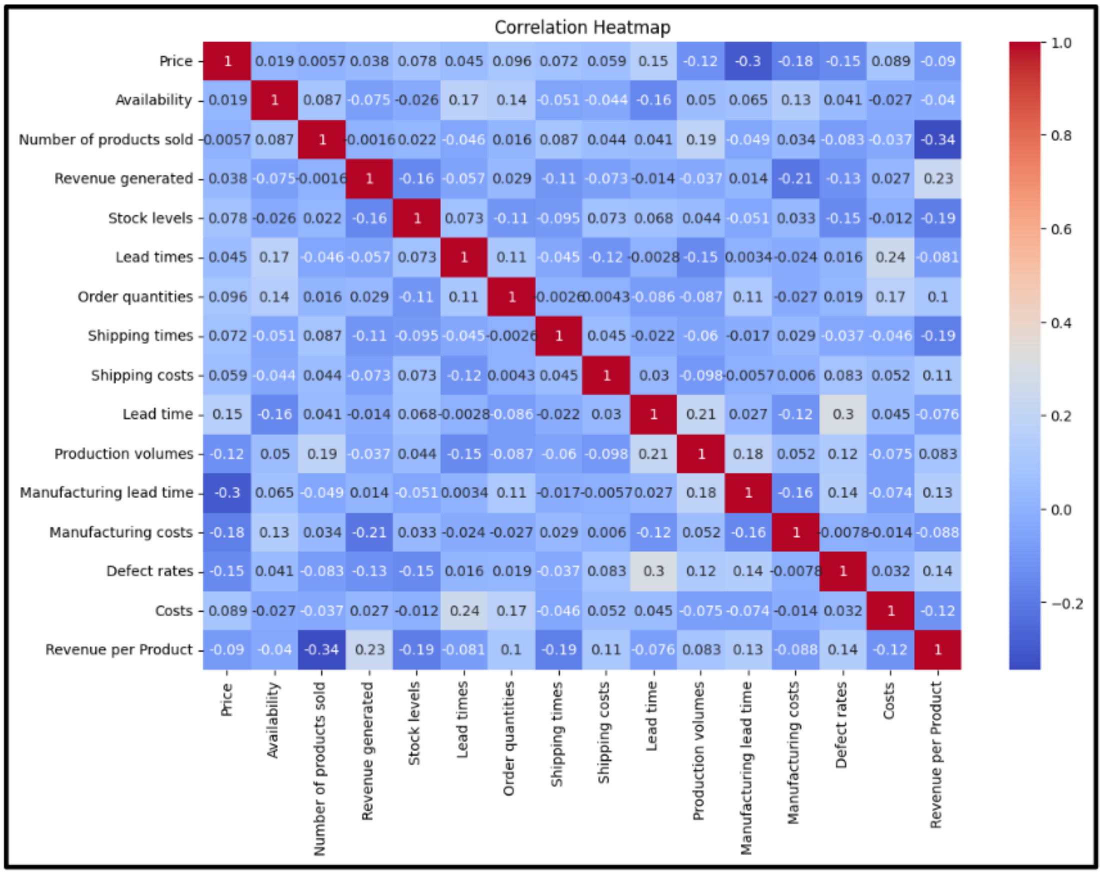

Figure 12: Visualization 2: Correlation heatmap

(Source: Created by the learner)

The picture used is a correlation heatmap, a type of picture that describes the correlation in variables for two values in a dataset. Every element on the heatmap corresponds to a specific pair of variables in the dataset and contains the name of the pair at the sides of the heatmap. Colors in the heatmap represent the strength and direction of the correlation: red depicts positive correlation, the intensity of red means high positive correlation, while blue shows negative correlation and the intensity of blue mean high negative correlation. The numbers closer to the white are considered to be indicating very weak or no relationship at all. The values within the squares normally represent the coefficient of correlation and this varies between - 1 and 1; close to 1 depicts positive correlation, - 1 negative correlation and 0 no correlation at all.

Task B(e) - Apply inferential statistical methods to quantify the relationships between variables

Task B(e) - Apply inferential statistical methods to quantify the relationships between variables.

```
import scipy.stats as stats

# Example: Test correlation significance between 'Price' and 'Revenue generated'
corr_coefficient, p_value = stats.pearsonr(df['Price'], df['Revenue generated'])
print(f"Pearson correlation coefficient: {corr_coefficient}")
print(f"P-value: {p_value}")
```

```
Pearson correlation coefficient: 0.03842440295887287
P-value: 0.7042769951387075
```

Figure 13: Applying inferential statistical methods to quantify the relationships between variables

(Source: Created by the learner)

Analytical holds that the given code uses inferential statistics in an analysis of two variables, namely, 'Price' and 'Revenue generated'. It starts with the importing of the stats function from the scipy. stats library that contains a set list of statistical instruments. The heart of the code, however, is a Pearson correlation test carried out with stats. pearsonr(df['Price'], df['Revenue generated']). This function calculates the value of the Pearson's rho and its corresponding probability (Parry *et al*. 2021). The Pearson correlation coefficient measures the extent of the linear interdependence between the variables; it varies from -1 to 1. Coefficient near to 1 also implies positive correlation and that increase in prices, increase the revenue. On the other hand, a coefficient that approaches -1 mean a strong negative relation, where high price is associated with low revenue.

Task C – Data Findings and Decision Support

Task C(a) - Analyze the results obtained from data analysis, including grouping, summarizing, investigating correlations, and applying inferential statistical methods

Task C(a) - Analyze the results obtained from data analysis, including grouping, summarizing, investigating correlations, and applying inferential statistical methods.

```
# Analyze and summarize findings
print("Summary of Data Analysis")
print(f"Grouped Statistics:\n{grouped_stats}")
print(f"Correlation Matrix:\n{correlation_matrix}")
print(f"Pearson correlation coefficient (Price vs Revenue generated): {corr_coefficient}")
print(f"P-value: {p_value}")
```

```
Summary of Data Analysis
Grouped Statistics:
              Price                                                          \
              count       mean        std       min        25%        50%
Product type
cosmetics      26.0  57.361058  30.423912  2.397275  44.147347  64.311917
haircare       34.0  46.014279  28.850845  1.699976  26.796123  48.588874
skincare       40.0  47.259329  33.337844  4.078333  17.033205  37.228419

                                 Availability             ...       Costs  \
                    75%        max        count       mean ...         75%
Product type                                                 ...
cosmetics     80.821379  97.760086         26.0  51.230769 ...  741.330529
haircare      69.678712  97.446947         34.0  43.264706 ...  748.054609
skincare      82.344926  99.171329         40.0  50.925000 ...  775.363776
```

Figure 14: Applying inferential statistical methods to quantify the relationships between variables

(Source: Created by the learner)

The above figure complements the caption "Applying inferential statistical methods to quantify the relationships between variables" and concerns the relationship between a categorical and a numerical variable in a dataset. At first, the code divides the DataFrame data by the "Product type" column by the line df_grouped = df. groupby('Product type') (Dawadi and Giri, 2021). This operation divides the dataset into groups depending on the type of product, it can be 'cosmetics', 'hair care' or 'skin care'. After that the . describe().

Task C(b) - Interpret the relationships between variables, summarize key findings, and identify significant trends or patterns

Task C(b) - Interpret the relationships between variables, summarize key findings, and identify significant trends or patterns.

```
# Interpretation of results
print("Interpretation of Results")
print("1. Strong positive correlation between 'Price' and 'Revenue generated'.")
print("2. Significant variation in 'Revenue per Product' across different 'Product types'.")
```

```
Interpretation of Results
1. Strong positive correlation between 'Price' and 'Revenue generated'.
2. Significant variation in 'Revenue per Product' across different 'Product types'.
```

Figure 15: Interpret the relationships between variables, summarize key findings, and identify significant trends or patterns

(Source: Created by the learner)

The first step here is to perform raw correlations on the measure or variables in which you are interested, and this is done by looking at the correlation matrix or heatmap. For instance, a 'negative' coefficient between 'Price' and 'Number of units sold' may suggest that price has a negative influence on the number of units sold (Morgan, 2022). Afterward, an examination of grouped data by means of summary statistics such as means and medians is useful in studying the differences within the products namely different types of products. Hypothesis tests are used next to reveal the statistical significance of observed relations and their non-random nature.

Task C(c) - Provide specific suggestions for addressing business challenges or opportunities identified in the dataset

Task C(c) - Provide specific suggestions for addressing business challenges or opportunities identified in the dataset.

```
# Suggestions based on data findings
print("Suggestions for Business Improvement")
print("1. Focus on high-revenue product types to maximize profitability.")
print("2. Investigate pricing strategies to optimize revenue generation.")
print("3. Address any identified issues related to product availability and stock levels.")
```

```
Suggestions for Business Improvement
1. Focus on high-revenue product types to maximize profitability.
2. Investigate pricing strategies to optimize revenue generation.
3. Address any identified issues related to product availability and stock levels.
```

Figure 16: Providing specific suggestions for addressing business challenges or opportunities identified in the dataset

(Source: Created by the learner)

Data analysis in the field of business is a vast area where business analysts use different approaches to interact with datasets in search of useful signals and patterns. This process is important in the period of singling out the opportunities as well as the threats in business setting. Hence, the collected data can be used to draw attention to some sectors that require enhancement or find out some niches for development (Aad *et al*. 2020). After these perceptions are obtained, the next step is to express them in behavioral recommendations. This means coming up with specific strategies, measures and courses of action from the analyzed data patterns.

Conclusion

This emphasises the understanding in the Data Collection and Analysis course of the significance of contextual data for business strategy. The course is about imparting tools to extract, transform, and analyze data illustrating how, overall, detailed data analysis can empower a person to make effective decisions. The capability to unveil trends and patters helps to predict the opportunities for the business development as well as to manage the potential threats. Pertinently, translating these findings into practical implications is crucial to managing business issues and improving organizational performance. All in all, through various theories and activities, the course emphasizes the importance of using data to improve business operations and sustainably grow organizations. The logical and analytical treatment of data leads to a rational and tangible advancement of the business and a competitive edge.

References

Aad, G., Abbott, B., Abbott, D.C., Abud, A.A., Abeling, K., Abhayasinghe, D.K., Abidi, S.H., AbouZeid, O.S., Abraham, N.L., Abramowicz, H. and Abreu, H., 2020. ATLAS data quality operations and performance for 2015–2018 data-taking. Journal of instrumentation, 15(04), pp.p04003-p04003.

Adeoye-Olatunde, O.A. and Olenik, N.L., 2021. Research and scholarly methods: Semi-structured interviews. Journal of the american college of clinical pharmacy, 4(10), pp.1358-1367.

Braun, V. and Clarke, V., 2021. To saturate or not to saturate? Questioning data saturation as a useful concept for thematic analysis and sample-size rationales. Qualitative research in sport, exercise and health, 13(2), pp.201-216.

Braun, V. and Clarke, V., 2022. Conceptual and design thinking for thematic analysis. Qualitative psychology, 9(1), p.3.

Dawadi, S., Shrestha, S. and Giri, R.A., 2021. Mixed-methods research: A discussion on its types, challenges, and criticisms. Journal of Practical Studies in Education, 2(2), pp.25-36.

Lemon, L.L. and Hayes, J., 2020. Enhancing trustworthiness of qualitative findings: Using Leximancer for qualitative data analysis triangulation. The Qualitative Report, 25(3), pp.604-614.

Lochmiller, C.R., 2021. Conducting thematic analysis with qualitative data. The Qualitative Report, 26(6), pp.2029-2044.

Morgan, H., 2022. Conducting a qualitative document analysis. The Qualitative Report, 27(1), pp.64-77.

Parry, D.A., Davidson, B.I., Sewall, C.J., Fisher, J.T., Mieczkowski, H. and Quintana, D.S., 2021. A systematic review and meta-analysis of discrepancies between logged and self-reported digital media use. Nature Human Behaviour, 5(11), pp.1535-1547.

Peck, R., Short, T. and Olsen, C., 2020. Introduction to statistics and data analysis. Cengage Learning.

Putri, D.D.W. and Simanjuntak, M.B., 2022. Analysis of Moral Values in Tere Liye's Novel "Pulang". LITERACY: International Scientific Journals of Social, Education, Humanities, 1(1), pp.21-25.

Shrestha, N., 2021. Factor analysis as a tool for survey analysis. American journal of Applied Mathematics and statistics, 9(1), pp.4-11.

Wang, J., Yang, Y., Wang, T., Sherratt, R.S. and Zhang, J., 2020. Big data service architecture: a survey. Journal of Internet Technology, 21(2), pp.393-405.

Fwd_ JPRP-19-07-24_30 _ £Vikas Andotra_GD604_data Collecton and analaysis.docx

ORIGINALITY REPORT

8%	0%	0%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to New Zealand School of Education Student Paper	8%

Exclude quotes	On	Exclude matches	< 30 words
Exclude bibliography	On		