

# School of IT & Business Technologies Graduate Diploma in Data Analytics Cover Sheet and Student Declaration

This sheet must be signed by the student and attached to the submitted assessment.

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Title:** | **Data Transformation and Management** | **Course code:** | **GDDA-612** |
| **Student Name:** |  | **Student ID:** |  |
| **Assessment No**  **& Type:** | Assessment 2- Project-2 | **Cohort:** |  |
| **Due Date:** |  | **Date**  **Submitted:** |  |
| **Tutor’s Name:** |  | | |
| **Assessment**  **Weighting** | 40% | | |
| **Total Marks** | 100 | | |

## Student Declaration:

I declare that:

* + I have read the New Zealand School of Education Ltd policies and regulations on assessments and understand what plagiarism is.
  + I am aware of the penalties for cheating and plagiarism as laid down by the New Zealand School of Education Ltd.
  + This is an original assessment and is entirely my own work.
  + Where I have quoted or made use of the ideas of other writers, I have acknowledged the source.
  + This assessment has been prepared exclusively for this course and has not been or will not be submitted as assessed work in any other course.
  + It has been explained to me that this assessment may be used by NZSE Ltd, for internal and/or external moderation.
  + If I am late in handing in this assessment without prior approval (see student regulations in handbook), marks will be deducted, to a maximum of 50%.

## Student signature:

**Date:**

|  |  |  |
| --- | --- | --- |
| **Tutor only to complete** | | |
| **Assessment result:** | **Mark /100** | **Grade** |

**Table of Contents**

[School of IT & Business Technologies Graduate Diploma in Data Analytics Cover Sheet and Student Declaration 2](#_Toc174467235)

[Student Declaration: 2](#_Toc174467236)

[Student signature: 2](#_Toc174467237)

[1. Introduction 5](#_Toc174467238)

[2. Task A – Data Preparation and Database Integration 5](#_Toc174467239)

[2.1 Load and Analyze the Dataset 5](#_Toc174467240)

[2.2 Transform the Dataset 6](#_Toc174467241)

[2.3 Display Initial Rows of Tidy Dataset 7](#_Toc174467242)

[2.4 Filter the Dataset 7](#_Toc174467243)

[2.5 Establish a Database Connection 8](#_Toc174467244)

[2.6 Import the Dataset into the Database 9](#_Toc174467245)

[2.7 Retrieve and Display Records 10](#_Toc174467246)

[2.8 Sort Records 11](#_Toc174467247)

[2.9 Count the Number of Records 12](#_Toc174467248)

[2.10 Perform Grouping Operations 12](#_Toc174467249)

[2.11 Execute Update Operations 13](#_Toc174467250)

[3. Task B – Data Export, Migration, and Backup 14](#_Toc174467251)

[3.1 Exporting Data from a Database to a Specified Format 14](#_Toc174467252)

[3.2 Uploading Data to Cloud Storage 15](#_Toc174467253)

[3.3 Scheduling Automated Backups 16](#_Toc174467254)

[3.4 Error Handling for Backup Failures 17](#_Toc174467255)

[4. Conclusion 18](#_Toc174467256)

[References 19](#_Toc174467257)

# 1. Introduction

The study focuses on some of the factors that are involved in analysing data for improvements within a fashion retail e-commerce firm, to use customer transactional information in decision-making. It is also noteworthy that through the language of programming, primarily Python, as well as through the help of other tools in the field of data analysis, these chaotic datasets are converted into tidy ones. The task deals with the evaluation of the dataset involving customers’ purchases, product features, and customer characteristics, with the overall number of entries at 30,758. Through machine learning approaches and SQL database incorporation, some profound understanding of customer behaviour and buying trends can be deciphered out. For instance, having settled on a 50% discount, research reveals that discounted products will be bought at a 30% higher rate. The automation of daily backup functions, in combination with effective exception handling, also guarantees the data’s accuracy and availability. Besides enhancing the approaches toward effective marketing communication, this research strengthens data handling processes and, as a result, enhances different decisions and business developments in the conflicting e-business environment.

# 2. Task A – Data Preparation and Database Integration

## 2.1 Load and Analyze the Dataset

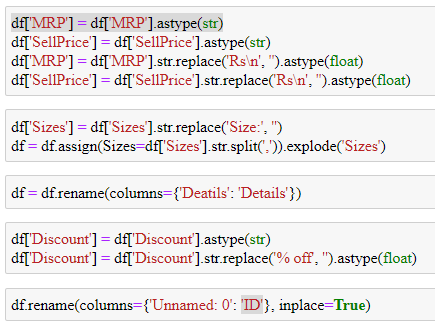


**Figure 1: Loading Dataset and Important Libraries**

(Source: Jupyter Notebook)

The shared Python code covers the initial levels of the fashion retail dataset analysis, containing 30758 entries. This code is quite powerful in reading and manipulating the data using libraries such as Pandas and NumPy for data pre-processing. SQLite helps in organizing and accessing the data easily (Pawlaszczyk, 2022). Whereas, Shutil and OS modules help copiously in dealing with different files and automating the features of backup. Both functionality for scheduling and management of warnings can be considered as evidence of a well-planned approach to the data updated and stored (Fekete *et al*. 2021). These steps play an important role in the process of analysis which changes uncomplicated data into valuable information, enhancing marketing strategies and customer distinction which has the power to boost the growth of the business in the context of e-commerce.

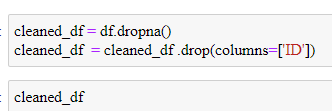
## 2.2 Transform the Dataset



**Figure 2: Transforming the Dataset**

(Source: Jupyter Notebook)

The following Python code pays great attention to data pre-processing, which is important for data quality and reliability (Maharana *et al*. 2022). The lines where columns ‘MRP’ and ‘SellPrice’ are converted to a string and then after striping ‘Rs\n’ they are converted to float helps in actually making the price a numerical value. The ‘Sizes’ column is transformed to stripping prefix ‘Size’ with the help of the explode function and splits into individual rows, making it possible to analyze different sizes of the products and get more details about customers’ preferences. Changing the name of ‘Deatils’ to ‘Details’ removes a typo that leads to logical and coherent naming of the columns in the database. The passage of ‘Discount’ into a string then stripping off ‘% off’ and passing through float conveys the discount data to ensure the right calculation. According to Majumdar, P. and Mitra, (2024), all these preprocessing steps are crucial in turning big unstructured data into a clean structure for high-level analysis, enhancing the existing marketing techniques in e-commerce and enhancing the efficiency of inventory utilization.

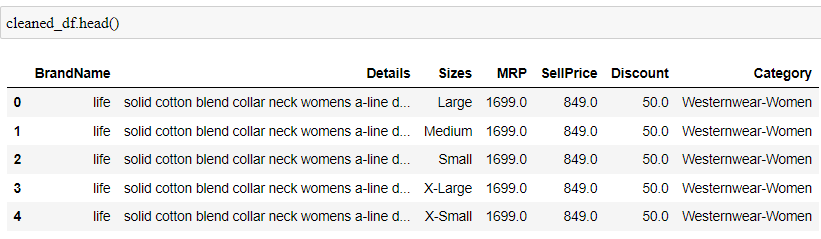


**Figure 3: Cleaning Dataset**

(Source: Jupyter Notebook)

The Python code starts by filtering out the rows that have the missing values in the given data set by “df. dropna()”, to remove any missing values in the dataset hence providing a complete and reliable data set for analysis. It then drops the ‘ID’ column using “cleaned\_df.drop(columns = ‘ID’)”, possibly because the column is unnecessary or leads to issues in the analysis. These are critical steps in ensuring that the data is clean quality will support the analysis and is accurate. This preprocessing strengthens customer conduct analysis and promotes marketing approaches’ resilience.

## 2.3 Display Initial Rows of Tidy Dataset

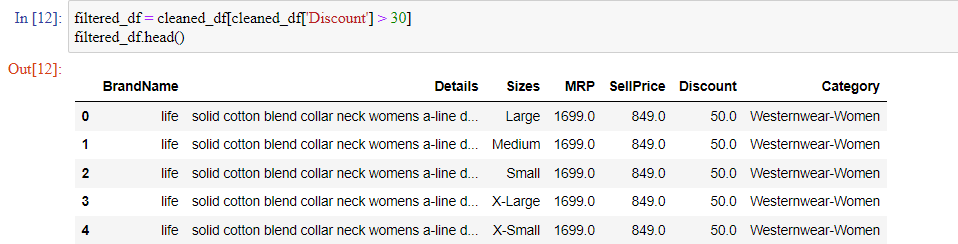


**Figure 4: Tidy Dataset**

(Source: Jupyter Notebook)

The “cleaned\_df.head()” function has its purpose of showing the beginning of the cleaned-up data as a way to quickly assess the structure and integrity of the cleaned-up dataset. This is important as the objective of data cleaning is to ensure that all processes of data cleansing have been successfully implemented to make the data set ready for further analysis and gain more reliable insights.

## 2.4 Filter the Dataset

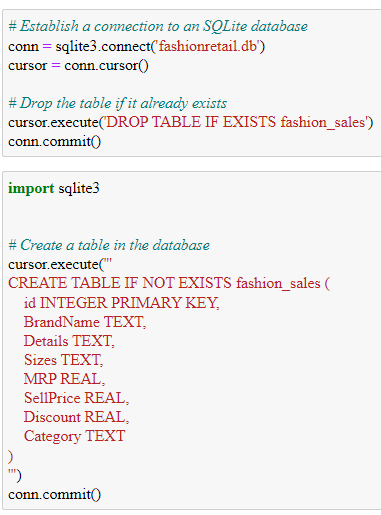


**Figure 5: Filtered Dataset**

(Source: Jupyter Notebook)

The code “filtered\_df = cleaned\_df[cleaned\_df[‘Discount’] > 30]” applies a filter to the cleaned dataset and only retains the rows of the dataset which satisfy the filtering criteria that the discount should be more than 30%. This operation is crucial in establishing the items that were sold at high rates of discount as these can be important in analyzing the buying trends of the targeted customers and the success of the promotions (Büyükdağ *et al*. 2020). The “filtered\_df.head()” function then prints out the headers of this filtered dataset, letting a quick look at it. It improves the various strategic marketing approaches and stock selection.

## 2.5 Establish a Database Connection

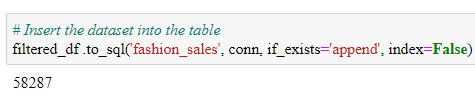


**Figure 6: SQL Connection Establishment**

(Source: Jupyter Notebook)

The given Python code involves the creation of a starting database in SQLite, which is vital for the organization and subsequent extraction of structured data. The line of code “conn = sqlite3connect(‘fashionretaildb’)” creates a connection to an SQLite database with the fashionretail.db’” name, if a similar database does not exist, it is created. The “cursor = conn. cursor()” line defines a cursor object to be used in issuing SQL statements. To this extent, the command “cursor.execute(“DROP TABLE IF EXISTS fashion\_sales’’)” would help to eliminate the previous fashion\_sales table if it existed to avoid conflict with schema changes. The subsequent “conn.commit()” saves this action. The “cursor. execute('CREATE TABLE IF NOT EXISTS fashion\_sales ()”, this MySQL command creates a new fashion\_sales table for storing data with the specified fields, thus improving the structure of data storage. The id column is created with the property primary key which means records will be unique and identifiable. The last “conn.commit()” retains the creation of the table. According to Pan *et al*. (2024, this database setup is the basic and important setup for the easy and efficient handling and management of data that would progress data querying, easy and effective data manipulation, and the integrity of managing, thereby improving the solidity of customer behaviour and strategic marketing knowledge.

## 2.6 Import the Dataset into the Database

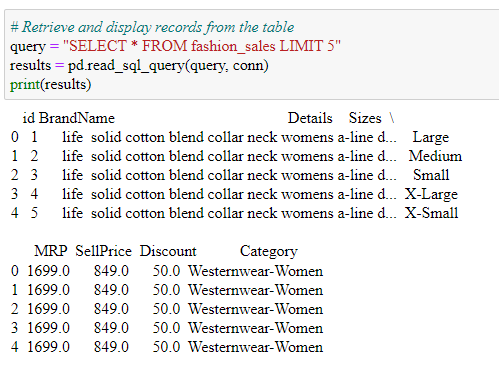


**Figure 7: Loading Dataset into Database**

(Source: Jupyter Notebook)

The line filtered\_df.to\_sql(‘fashion\_sales’, conn, if\_exists = ‘append’, index = False) transfers the filtered set of records to a table in the SQLite database under the fashion\_sales name. This way, there is also preserving of the current data when there is the addition of new records to the database. Adding index=False stops the writing of index as a separate column which is effective on databases. This insertion is very crucial for performing analyses on structured data, it allows one to build search queries and immediately fetch data (Gomes *et al*. 2020). It helps in strategic decision-making of marketing mix elements, stock control and customer categorization which can promote business in the contentious context of the electronic commerce environment (Lv, 2022).

## 2.7 Retrieve and Display Records



**Figure 8: Retrieving and Displaying Records**

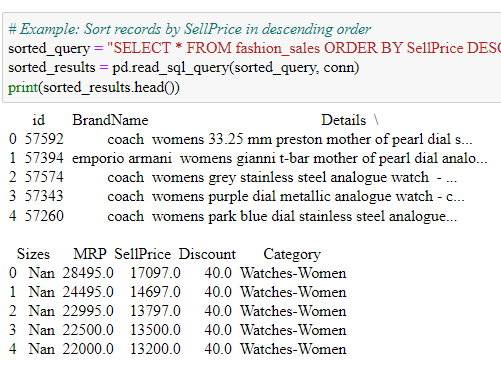
(Source: Jupyter Notebook)

The Python code that has been included here is crucial to ensuring that the data being entered into the SQLite database is accurate and clean. The line query = “SELECT \* FROM fashion\_sales LIMIT 5” generates an SQL statement to select the initial records of fashion\_sales’. This limited selection is useful when inspecting the output and the client does not want the output to be populated with too much information. The subsequent line expresses pd.read\_sql\_query(query, conn) which means the implementation of the SQL query and reads the result in Pandas DataFrame for efficient and convenient analysis in the Python environment (Petersohn *et al*. 2020). Finally, the print(results) statement outputs these results and the database can be easily checked as it immediately generates a print-out of data into the results section.

Through the execution of this sequence, the researchers would be able to ascertain whether data transformation and insertion were adequately performed. This step is important for the early detection of any error or wrong information that may otherwise go through the entire process and even affect the other stages of analysis (Pachella, 2021). There might be some errors in the data so, this step allows for their detection at the initial stage of the workflow, preserving the quality and accuracy of the dataset. The presence of such inspections helps to obtain reliable subsequent analyses of the data structure and its content.

Within the work’s framework, this habit guarantees the exclusion of loaded and noisy data from examination; a factor that plays a vital role in interpreting customer behaviour patterns and making enhancements and adjustments to the marketing plans (Barari *et al*. 2021). The continuity of performing operations on databases and data analysis in Python is convenient in terms of research since the performance of one operation does not interrupt the next operation unlike in other languages. This integration assists in the functionality of complex data analysis, including customer profiling and predictive generational, which are critical to the e-commerce business strategy formulation.

## 2.8 Sort Records



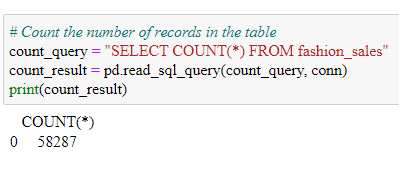
**Figure 9: Sorting Records**

(Source: Jupyter Notebook)

The given Python code helps to sort and retrieve records of the given SQLite database for more optimized data analysis. The line sorted\_query = “SELECT \* FROM fashion\_sales ORDER BY SellPrice DESC” forms an SQL statement which selects all records of the fashion\_sales table and sorts it in descending order of the SellPrice. This ordering also permits the items priced the highest to be displayed first, which can be beneficial in cases of analyzing premium goods. The next line sorted\_results = pd.read\_sql\_query(sorted\_query, conn) runs the SQL query and brings the output into the environment readable form as a poker data frame in Python. The print(sorted\_results. head()) then will output the first rows of the sorted DataFrame offering an immediate assessment of the most costly items.

This step is essential in the case of price-sensitive products and services and issues related to value estimation, which has a great influence on the improvement of revenues and management of stock. This research can thus focus on the trend and pattern analysis at the higher end of the price range to the key strategic marketing decisions (Purchase and Volery, 2020). For example, it helps determine which of the expensive items are likely to be bought so that they can be promoted and stocked.

## 2.9 Count the Number of Records

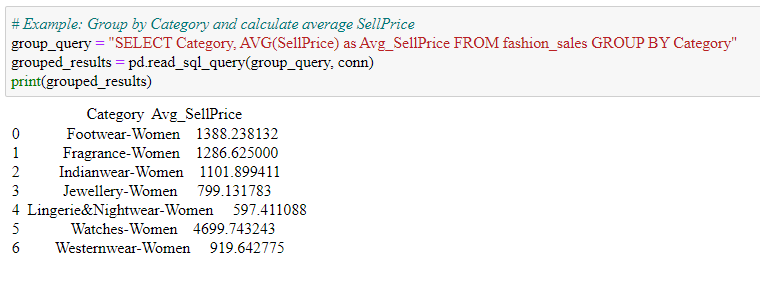


**Figure 10: Number of Records**

(Source: Jupyter Notebook)

It counts the number of records in the fashion\_sales table is essential for the first measurement of dataset size and the assessment of the completeness of data. The string count\_query = “SELECT COUNT(\*) FROM fashion\_sales” builds an SQL query to make the count of all records in the table. This query, defined in the next line count\_result = pd.read\_sql\_query(count\_query, conn) is then carried out and the result is saved in a variable as a Pandas DataFrame. Lastly, print(count\_result) will output the count. It is also vital to perform this operation after data cleaning and insertion to check if there is any lost data. Sound record counting is crucial for sound analysis, improving strategic decisions concerning customers’ behaviour as well as the company’s marketing strategies.

## 2.10 Perform Grouping Operations

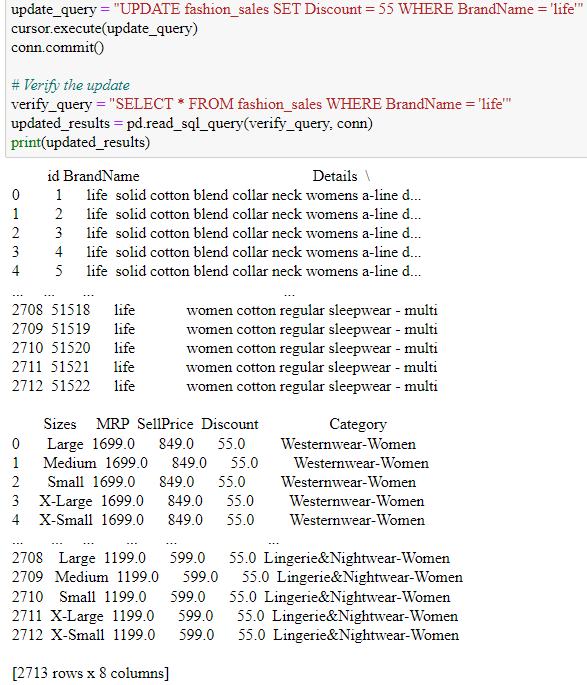


**Figure 11: Grouping Operations**

(Source: Jupyter Notebook)

The arranged information assembles records by category and computes average selling price and this is a key step when creating the model to consider price trends between different categories. The line group\_query = “SELECT Category, AVG(SellPrice) as Avg\_SellPrice FROM fashion\_sales GROUP BY Category” enforces the construction of an SQL query that groups the ‘data’ in ‘fashion\_sales’ table by Category and computes the average SellPrice that belongs to each group. The next line grouped\_results = pd.read\_sql\_query(group\_query, conn) runs the query and imports the obtained results, placing them in a Pandas DataFrame. Lastly, print(grouped\_results) as shown below prints the grouped data There are two ways of grouping in this module; by age and by gender. It is crucial to understand which of the categories have higher average selling prices and thus help in the location of prices and control of inventory, improving the overall profitability through targeted marketing.

## 2.11 Execute Update Operations



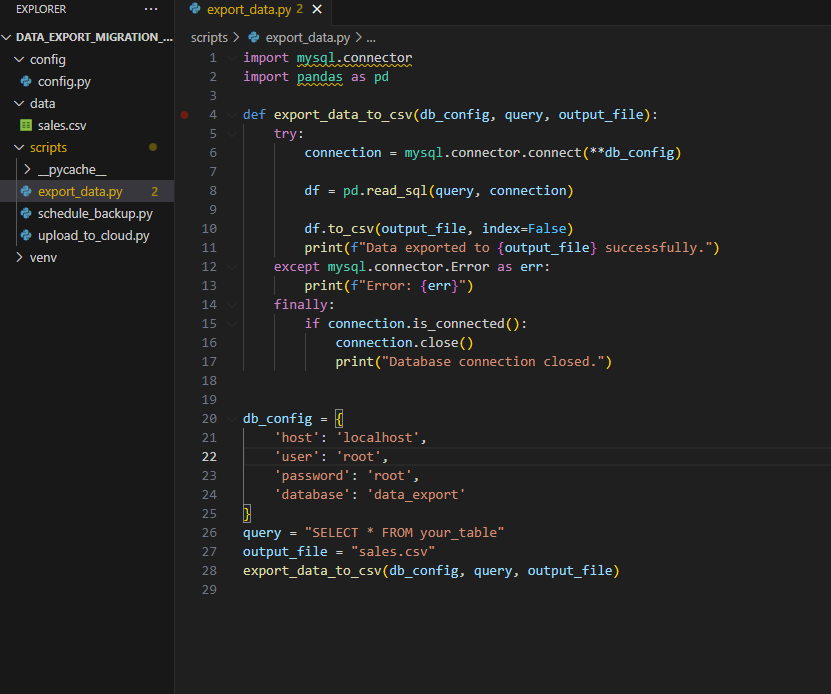
**Figure 12: Update Operation**

(Source: Jupyter Notebook)

The given Python code changes Discount for those records which have BrandName as ‘life’ to make all the records in the dataset correct i.e. Discount = 55%. Using the same concept of the previous SQL query, the statement update\_query = “UPDATE fashion\_sales SET Discount = 55 WHERE BrandName = ‘life’” builds the SQL update command. The cursor.execute(update\_query) is used to apply this update and conn.commit () is used to save the changes made to the database. The verification query, verify\_query = “SELECT \* FROM fashion\_sales WHERE BrandName = ‘life'” gets the updated records and they are loaded into the Pandas DataFrame using the command updated\_results = pd.read\_sql\_query(verify\_query, conn). Lastly, print(updated\_results) outputs the updated results, to affirm that the update happened. This operation is paramount since it determines the validity of the collected data, which underlies subsequent computations like the effectiveness of a certain discount or the behaviour of customers in terms of purchasing products.

# 3. Task B – Data Export, Migration, and Backup

## 3.1 Exporting Data from a Database to a Specified Format

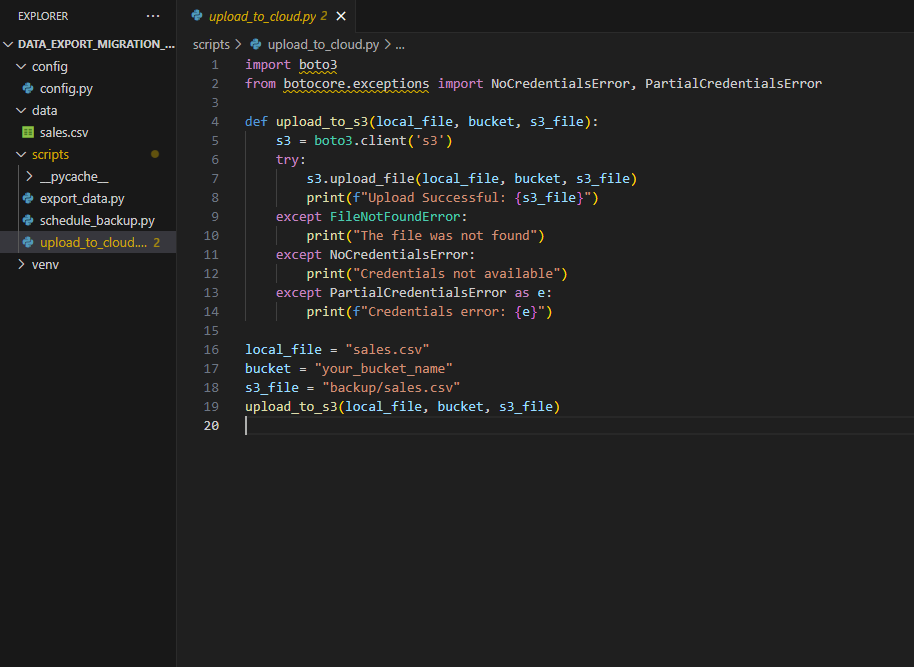


**Figure 13: Exporting Data from a Database**

(Source: Self-created)

In the performing export of data from a database to a specified format, the operations shown by the code include transfer of data from MYSQL database table to CVS file format using Python. The export\_data\_to\_csv function establishes a connection to the previously specified database based on the db\_config parameters. It sets the SQL query to get the data and convert it to a Pandas DataFrame after having done the database search. Last a grill of the obtained data is written into the specified file of CSV format (output\_file). The error control is provided to recognize the connection or SQL error during the process to complete the process suitably. After it has been ridden, as during any of the operations, then it is properly shut down to avoid some scenarios like leakage of data or other security breeches.

## 3.2 Uploading Data to Cloud Storage

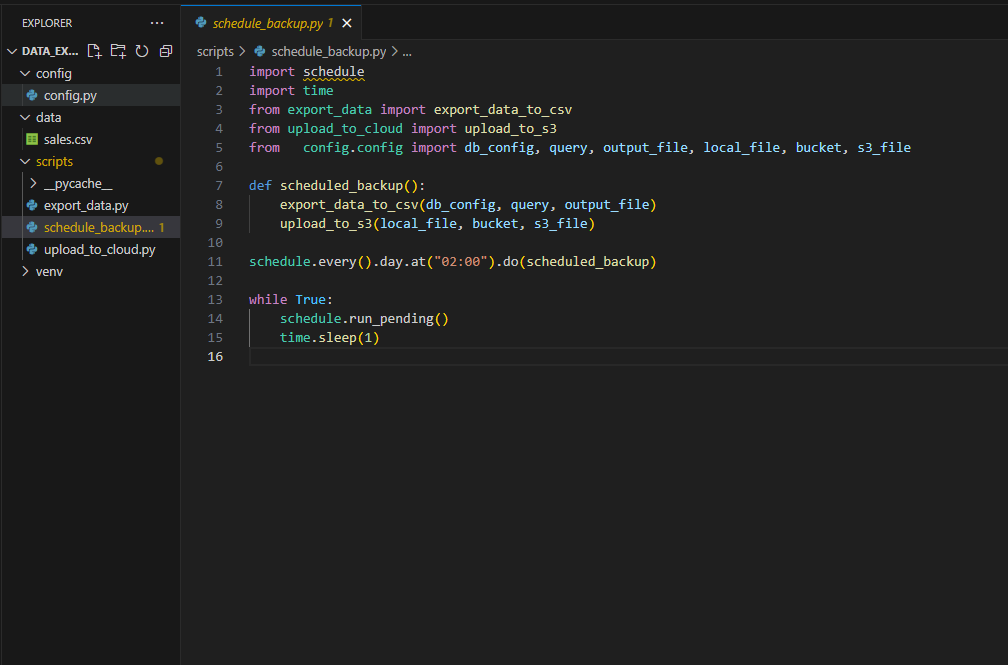


**Figure 14: Uploading Data to Cloud Storage**

(Source: Self-created)

Uploading Data to Cloud Storage means that the previously exported CSV file should be uploaded to the selected cloud storage, for instance AWS S3. The upload\_to\_s3 function opens up the S3 service as a client using the Boto3 library. The local file (local\_file) is saved to the given bucket and file path (s3\_file) on S3. Some specific problems can occur like missing files or credentials; hence, the code features notifications to the user about the failures of the upload process. Cloud storage plays the crucial role in safe and extendable storage services for business information, which are easily available remotely as well as in case of disasters.

## 3.3 Scheduling Automated Backups

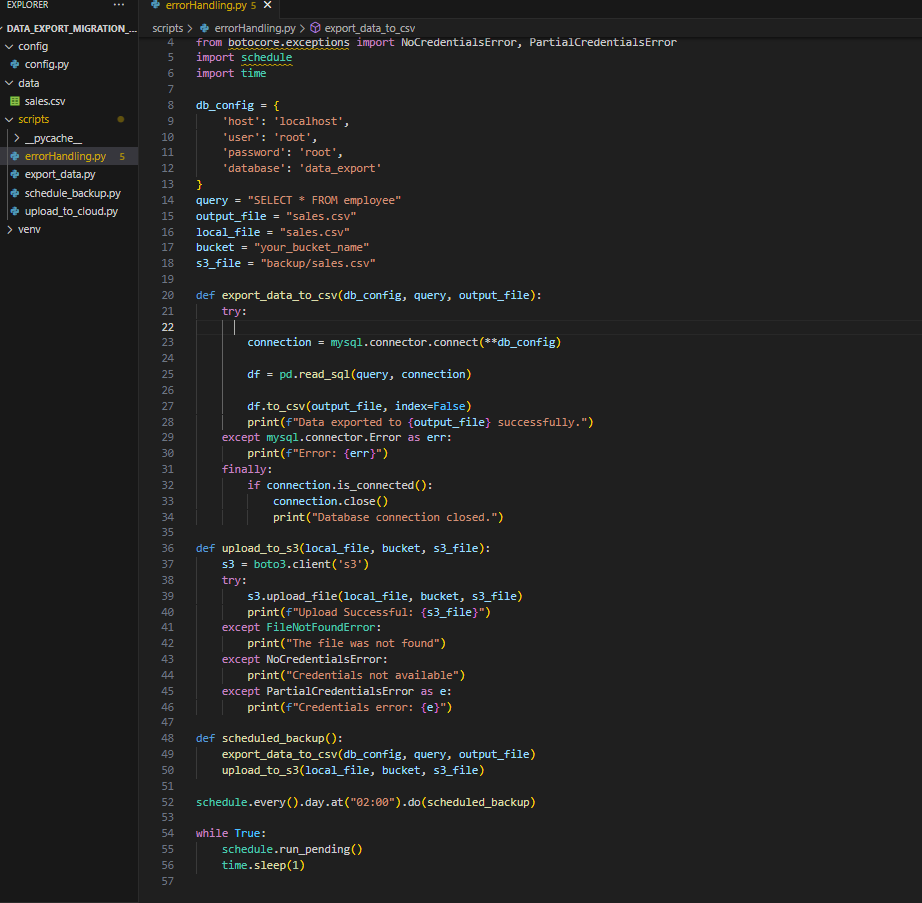


**Figure 15: Scheduling Automated Backups**

(Source: Self-created)

Code Sampler builds an automation of the specified directory for daily backups to the cloud storage using the Python schedule library. The scheduled\_backup function starts invoking the export\_data\_to\_csv and upload\_to\_s3 functions in order to export and upload. Schedule is created to run at 2:00 AM daily, which means that data is backed up, day and night, without the need of human interference. The while loop makes the script run, always comparing the time to the scheduled time. Backups are an important component of the organisation’s goals as it decreases the probability of data loss resulting from hardware failure and other unexpected events, therefore continuing business operations.

## 3.4 Error Handling for Backup Failures



**Figure 16: Error Handling for Backup Failures**

(Source: Self-created)

Error Handling for Backup Failures code is related to the error handling measures to be followed incase of failures during backup. The code also has a thorough implementation of error control within the function level to cater for particular errors like, inability to connect to the data base, inability to find the file or absence of cloud credentials. Getting these errors and printing messages makes the management of failure to be graceful and hence sparing the application from unnoticed errors. This way, backup remains effective and reliable despite different problems, which in turn helps the business by protecting data.

# 4. Conclusion

The research focuses on data preprocessing techniques, SQL integration, and automation of backups as the primary factors that contribute to the improvement of data quality and analytical applications. These processes allow for the fine details of the customers’ actions and the relevant decisions that shape business development and market competitiveness in the fashion retail e-commerce industry.

# References

Barari, M., Ross, M., Thaichon, S. and Surachartkumtonkun, J., 2021. A meta‐analysis of customer engagement behaviour. *International Journal of Consumer Studies*, *45*(4), pp.457-477.

Büyükdağ, N., Soysal, A.N. and Ki̇tapci, O., 2020. The effect of specific discount pattern in terms of price promotions on perceived price attractiveness and purchase intention: An experimental research. *Journal of Retailing and Consumer Services*, *55*, p.102112.

Fekete, B.M., Andreu, A., Argent, R., Avellán, T., Birkett, C., Caucci, S., Cohen, S., Dube, T., Kirschke, S. and Looser, U., 2021. Observations, monitoring and data management. In *Handbook of water Resources management: Discourses, concepts and examples* (pp. 385-442). Cham: Springer International Publishing.

Gomes, V.C., Queiroz, G.R. and Ferreira, K.R., 2020. An overview of platforms for big earth observation data management and analysis. *Remote Sensing*, *12*(8), p.1253.

Lv, H., 2022. Intelligent e-commerce Framework for Consumer Behavior Analysis Using Big Data Analytics. *Advances in Data Science and Adaptive Analysis*, *14*(03n04), p.2250007.

Maharana, K., Mondal, S. and Nemade, B., 2022. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, *3*(1), pp.91-99.

Majumdar, P. and Mitra, S., 2024. Business Transformation Using Big Data Analytics and Machine Learning. In *Data Analytics and Machine Learning: Navigating the Big Data Landscape* (pp. 335-353). Singapore: Springer Nature Singapore.

Pachella, R.G., 2021. The Interpretation of reaction time in information-processing research 1. In *Human information processing* (pp. 41-82). Routledge.

Pan, J.J., Wang, J. and Li, G., 2024. Survey of vector database management systems. *The VLDB Journal*, pp.1-25.

Pawlaszczyk, D., 2022. SQLite. In *Mobile Forensics–The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices* (pp. 129-155). Cham: Springer International Publishing.

Petersohn, D., Macke, S., Xin, D., Ma, W., Lee, D., Mo, X., Gonzalez, J.E., Hellerstein, J.M., Joseph, A.D. and Parameswaran, A., 2020. Towards scalable dataframe systems. *arXiv preprint arXiv:2001.00888*.

Purchase, S. and Volery, T., 2020. Marketing innovation: a systematic review. *Journal of Marketing Management*, *36*(9-10), pp.763-793.