**SES\_Survey\_1\_DataPush**

**Objective**

The objective of this report is to provide a detailed explanation of the code chunk that cleans and transforms the **"Studentsurvey.csv"** data for later use in BigQuery. This report will explain the specific steps taken in the code, as well as the functions and libraries used.

**Methodology**

The code chunk uses Python programming language and the following libraries:

**pandas**: for creating and manipulating DataFrames

**fuzzywuzzy**: for fuzzy string matching

**google-auth**: for authenticating the user in Google Colab

**google-cloud-bigquery**: for pushing data to BigQuery

The code follows the following steps:

**Step 1: Importing and Authenticating**

The code first authenticates the user in Google Colab using the google-auth library. The necessary libraries are then imported, including pandas and fuzzywuzzy.

**Step 2: Creating a DataFrame(df)**

Next, the code creates a Pandas DataFrame by reading the CSV file uploaded in the previous step. It then renames some of the columns and drops some of the columns that are not required.

**Step 3: Data Cleaning**

The code performs some data cleaning by applying the following transformations:

**State**: The code uses the fuzzywuzzy library to validate and format the state names. The **is\_valid\_state ()** function is defined and applied to the State column to find a valid state name based on the similarity score with the name entered by the user in the survey. If the similarity score is above 80, it returns the correct state name; otherwise, it returns NaN.

**Field of Experience:** The code fills missing values in the Field\_of\_Exp column with "No Experience" and groups the remaining values into broad categories such as **"Information Technology", "Finance & Banking", "Data Analytics", "Sales & Marketing", "Manufacturing", "Healthcare”, “Media/Entertainment", "Education", "Engineering", "Customer Service", "Project Management", and "Human Resources"**.

**Zip code:** The code formats the **Zip\_code** column by removing any non-alphanumeric characters, selecting the first six alphanumeric characters, padding the resulting string with spaces to make it six characters long, and splitting the string into two parts separated by a space. The format\_zip() function is defined to perform this transformation, and it is applied to the Zip\_code column of the DataFrame using the apply() method. The output is a DataFrame with the Zip\_code column formatted in the desired way.

**Handling Missing Values:** The code replaces all the missing values in the **Services\_Used, Feedback, Accm\_info\_source, Yrs\_Exp, and State** columns with the string 'None'. This transformation is performed using the replace() method of the Pandas DataFrame object, with np.nan (Numpy's representation of missing values) as the value to be replaced and 'None' as the replacement value. The resulting DataFrame has no missing values in the specified columns. This step helps ensure that the data is consistent and complete for analysis.

**Step 4: Creating a BigQuery Table and Inserting Data into it using Python.**

In this step, we are using the Google Cloud BigQuery API to connect to our BigQuery database and create a new table named **"Base\_Survey\_Table"** if it does not already exist. We define the schema for the table, which includes the column names and data types.

Next, we load the survey data stored in a pandas dataframe **(df)** into the new table using the **load\_table\_from\_dataframe()** method. We set the write disposition to **WRITE\_TRUNCATE**, which overwrites any existing data in the table, and the source format to **CSV**. Finally, we execute the load job using the **result()** method to insert the data into the table.

**Step 5: Breaking Base\_Survey\_Table to 7 other Tables**

Capstone Project involves breaking the Base\_Survey\_Table into 7 separate tables in order to organize and analyze the data more efficiently. To achieve this, the above SQL code is defined to create tables for each category of data and insert the relevant data from the **Base\_Survey\_Table**.

The SQL code first drops any existing tables with the same names as the ones to be created, to ensure that the tables are created fresh. It then creates a table for each category of data: **Academic, Academic\_feedback, Accommodation, Accommodation\_feedback, Demographics, Feedback, and Services.**

For each table, the appropriate columns are specified, and their data types defined. The SQL **INSERT INTO** statement is then used to populate each table with the relevant data from the **Base\_Survey\_Table**.

Once the SQL code has been defined, a query job is executed to run the code and create the tables. The result of the query job is not important, as the code itself is what creates and populates the tables.

**Step 6: Combining Academic and Demographic Data and Loading to Google BigQuery**

This step involves combining two tables - Demographics and Academic\_feedback - to create a new table called AcaDemo. The new table contains information on academic scores and demographic information for each ID. The code uses the pandas\_gbq library to read data from Google BigQuery for the two tables.

Next, the Academic\_feedback table is transformed using the pd.melt function, which pivots the table so that the Score\_Type column is melted down and new rows are created for each unique score type. The resulting table is then merged with the Demographics table using the pandas merge function and sorted by ID.

After combining the data, the code connects to the BigQuery project and drops the existing AcaDemo table if it exists. It then defines the schema for the new table using the bigquery.SchemaField function. If the table exists, the code gets the existing table; otherwise, it creates a new table with the defined schema. Finally, the data is loaded into the new table using the bigquery.load\_table\_from\_dataframe function.

**Step 7: Creating Academic and Accommodation Score Tables**

In this step, we are creating two new tables in Google BigQuery to store the academic and accommodation scores of students. The academic and accommodation scores are calculated using the weights assigned to various feedback parameters and stored in the "Acd\_score" and "Acc\_score" columns of the respective tables. The steps involved in creating the academic and accommodation score tables are as follows:

1. Reading the "Academic\_feedback" and "Accommodation\_feedback" tables from Google BigQuery using the pandas\_gbq package.
2. Copying the "Academic\_feedback" and "Accommodation\_feedback" tables to new DataFrame variables "af" and "acf" respectively, and mapping the feedback labels to numerical values.
3. Defining weights for each feedback parameter for both tables.
4. Calculating the academic and accommodation scores for each student by multiplying the score of each feedback parameter with its weight and adding all the scores.
5. Creating two new DataFrames "Academic\_score" and "Accommodation\_score" with "ID" and "Acd\_score" or "Acc\_score" columns respectively.
6. Connecting to the BigQuery project using the Google Cloud client library.
7. Dropping the existing "Academic\_score" and "Accommodation\_score" tables (if any).
8. Defining the schema for the new tables.
9. Checking if the tables exist in BigQuery. If not, creating new tables with the defined schema.
10. Inserting the data from the "Academic\_score" and "Accommodation\_score" DataFrames into the BigQuery tables.
11. Reading the data from the newly created tables to verify the successful push of data.

The above code performs all these steps and creates two new tables named "Academic\_score" and "Accommodation\_score" in the "Capstone\_Project" dataset of the "surveyproject-378222" project in Google BigQuery. The new tables have two columns - "ID" and "Acd\_score" or "Acc\_score" respectively. "ID" is the unique identifier for each student or respondent, and "Acd\_score" or "Acc\_score" is their academic or accommodation score calculated using the weights assigned to various feedback parameters. These tables will be useful in analyzing the survey results.