	Point_de_Vente   Numero_TPV   Numero_Transaction   Date_Transaction   Heur   Typologie_Magasin_os   Numero_Fidelit   Type_de_Vente   Univers_Produit   Segment_Produit     Fedas_Numero   Fedas_Libelle   Oilege_Magasin_os   PDV-id-00562   TPV_783   TID00002834579   31/01/2022   16:00:00   Typologie_Magasin_os   N_2563125   TV3   CL1_9   CL2_51     Fedas_Num2138   Fedas_Lib522   TV3   PDV-id-00240   TPV_769   TID00002759437   31/01/2022   15:00:00   Typologie_Magasin_os   N_1420193   TV3   CL1_9   CL2_51     Fedas_Num2138   Fedas_Lib522   TV3   TV3   TV3   TV3   TV3   TV3   TV3   TV4   TV3   TV4   TV
: [	## 1000002957223   17/03/2022   12:00:00 Typologie_Magasin_05   NaN TV3   CL1_10   CL2_72   FedasNum500 FedasLib1337   ## 1000002957223   17/03/2022   17:00:00 Typologie_Magasin_05   NaN TV3   CL1_10   CL2_72   FedasNum500 FedasLib1337   ## 1000005648532   14/03/2022   17:00:00 Typologie_Magasin_05   NaN TV3   CL1_10   CL2_72   FedasNum500 FedasLib1337   ## 10000025648532   14/03/2022   17:00:00 Typologie_Magasin_05   NaN TV3   CL1_10   CL2_72   FedasNum500 FedasLib1337   ## 10000025648532   14/03/2022   17:00:00 Typologie_Magasin_05   NaN TV3   CL1_10   CL2_72   FedasNum500 FedasLib1337   ## 10000025648532   14/03/2022   17:00:00 Typologie_Magasin_05   NaN TV3   CL1_10   CL2_72   FedasNum500 FedasLib1337   ## 10000025648532   TPV_10
	Typologie Magasin Numero_Fidelite Type_de_Vente Univers_Produit \ 0 Typologie Magasin 05
	Printing duplicate rows in a DataFrame  duplicate_rows = df[df.duplicated()] print(duplicate_rows)  424228 Point_de_Vente   Numero_Transaction   Date_Transaction   16/82/2022   16/82/2022   424228   Point_de_Vente   Numero_Transaction   Date_Transaction   16/82/2022   16/82/2022   424228   Numero_Transaction   Date_Transaction   Date_Transaction   Date_Transaction   16/82/2022   424228   Numero_Transaction   Date_Transaction
	Point_de_Vente   Numero_TPV   Numero_Transaction   Date_Transaction   Value
į	Point_de_Vent   Numero_TPV   Numero_Transaction   Date_Transaction
	print(df.shape)  (543060, 22)  Calculating Missing Values for Each Column  missing_values = df.isnull().sum() print(missing_values)  Point_de_vente Numero_TPV
]: [	Segment_Produit 240 Famille_Produit 240 Sous_Famille_Produit 240 Fedas_Numero 0 Fedas_Libelle 0 Cible_Genre_Age 0 Modele_Couleur_Ref 0 Modele_Couleur_Libelle 0 Type_de_vente_NPS 0 Quantite_Vendue 0 CA_Net_TT 0 CA_Net_TT 0 Marge_Nette_Magasin 0 dtype: int64  Filling Missing Values in 'Numero_Fidelite' Column   df['Numero_Fidelite'].fillna('DefaultValue', inplace=True)  C:\Users\HP\anaconda3\lib\site-packages\pandas\core\generic.py:6392: SettingWithCopyWarning:
]: [ ]: [	A value is trying to be set on a copy of a slice from a DataFrame  See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy return selfupdate_inplace(result)  Filling Missing Values in 'Numero_TPV' Column  df['Numero_TPV'].fillna('DefaultValue', inplace=True)  Filling Missing Values in 'Segment_Produit' Column  df['Segment_Produit'].fillna('DefaultValue', inplace=True)  Filling Missing Values in 'Famille_Produit' Column  df['Famille_Produit'].fillna('DefaultValue', inplace=True)
] ]: [ ]: [ ]: [	Filling Missing Values in 'Sous_Famille_Produit' Column  df['Sous_Famille_Produit'].fillna('DefaultValue', inplace=True)  Filling Missing Values in 'Univers_Produit' Column  df['Univers_Produit'].fillna('DefaultValue', inplace=True)  Installing Necessary Libraries for Data Processing and Excel Operations  pip install pandas openpyx1  Requirement already satisfied: pandas in c:\users\hp\anaconda3\lib\site-packages (1.3.4)
	Requirement already satisfied: openpyxl in c:\users\hp\anaconda3\lib\site-packages (3.0.9) Requirement already satisfied: numpy>=1.17.3 in c:\users\hp\anaconda3\lib\site-packages (from pandas) (2.8.2) Requirement already satisfied: pytbno-dateutil>=2.7.3 in c:\users\hp\anaconda3\lib\site-packages (from pandas) (2.8.2) Requirement already satisfied: pytz>=2017.3 in c:\users\hp\anaconda3\lib\site-packages (from pandas) (2.8.2) Requirement already satisfied: et-xmlfile in c:\users\hp\anaconda3\lib\site-packages (from pandas) (2.8.2) Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-packages (from pandas) (2.23.3) Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0) Note: you may need to restart the kernel to use updated packages.  WARNING: Ignoring invalid distribution - (c:\users\hp\anaconda3\lib\site-packages)  WARNING: Ignoring invalid distribution - (c:\users\hp\anaconda3\lib\site-packages)  WARNING: Ignoring invalid distribution - (c:\users\hp\anaconda3\lib\site-packages)  WARNING: Ignoring invalid distribution - (c:\users\hp\anaconda3\lib\site-packages)  WARNING: Ignoring invalid distribution - (c:\users\hp\anaconda3\lib\site-packages)  WARNING: Ignoring invalid distribution - (c:\users\hp\anaconda3\lib\site-packages)
 	df2 = pd.read_excel(file_path, engine='openpyx1')  Displaying the dataframe  df2  CODE_MODELE_COULEUR_ACTUEL SIGNATURE_PRODUCT 0 MCR1 True 1 MCR2 True 2 MCR3 True 3 MCR4 False 4 MCR5 False 4 MCR5 False 5 MCR5 False 6 MCR5 False
; ; ; ; ;	209995         MCR209996         False           209996         MCR209997         False           209997         MCR209998         False           209998         MCR209999         False           20000 rows × 2 columns         Checking for duplicates in the dataframe           Inum_duplicates = df2.duplicated().sum() print(f"Number of duplicate rows: {num_duplicate rows
]: [ (	checking the dimensions of the dataframe  print(df2.shape)  (210000, 2)  Calculating Missing Values for Each Column  missing_values = df2.isnull().sum() print(missing_values)  CODE_MODELE_COULEUR_ACTUEL 0 SIGNATURE_PRODUCT 0 dtype: int64
]:[	Previewing the First Rows of the DataFrame   print(df2.head())  CODE_MODELE_COULEUR_ACTUEL SIGNATURE_PRODUCT 0 MCR1 True 1 MCR2 True 2 MCR3 True 3 MCR4 False 4 MCR5 False  Checking for the duplicates  num_duplicates = df2.duplicated().sum() print(f"Number of duplicate rows: {num_duplicates}")  Number of duplicate rows: 0
]: [	checking for the missing values for each column  missing_values = df2.isnull().sum() print(missing_values)  CODE_MODELE_COULEUR_ACTUEL
	Heure TEXT NOT NULL, or TIME if consistently in HH:MM:SS format Typologie_Magasin VARCHAR(30) NOT NULL, Numero_Fidelite CHAR(15) NOT NULL, Type_de_Vente CHAR(3) NOT NULL, Univers_Produit CHAR(10), Segment_Produit CHAR(10), Famille_Produit CHAR(10), Sous_Famille_Produit CHAR(10), Fedas_Numero CHAR(20), Fedas_Libelle CHAR(20), Cible_Genre_Age CHAR(4), Modele_Couleur_Ref VARCHAR(30), Modele_Couleur_Libelle VARCHAR(30), Type_de_vente_NPS CHAR(4), Quantite_Vendue INTEGER NOT NULL, CA_Net_TIC INTEGER NOT NULL, Marge_Nette_Magasin INTEGER NOT NULL)""")
	OperationalError Traceback (most recent call last)  -\Apphata\Local\Temp/ipykernel_11228/2924456683.py in <module>&gt; 1 cursor.execute ("""CREATE TABLE Transactions (  2</module>
	OperationalError Traceback (most recent call last)  ~\AppData\Local\Temp/ipykernel_11228/3780574549.py in <module>&gt; 1 cursor.execute("""  2 CREATE TABLE Products ( 3 CODE_MODELE_COULEUR_ACTUEL VARCHAR(50) NOT NULL PRIMARY KEY, 4 SIGNATURE_PRODUCT INTEGER NOT NULL CHECK (SIGNATURE_PRODUCT IN (0, 1))  5 )  OperationalError: table Products already exists  Database Update: Appending to "Transactions table"  df.to_sql('Transactions', connection, if_exists='append', index=False)</module>
] : [   	Listing Column Names of the DataFrame  print(df.columns.tolist())  ['Point_de_Vente', 'Numero_TPV', 'Numero_Transaction', 'Date_Transaction', 'Heure', 'Typologie_Magasin', 'Numero_Fidelite', 'Type_de_Vente', 'Univers_Produit', 'Segment_Produit', 'Sous_Famille_Produit', 'Fedas_Numero', 'Fedas_Libelle', 'Cible_Genre_Age', 'Modele_Couleur_Ref', 'Modele_Couleur_Libelle', 'Type_de_vente_NPS', 'Quantite e', 'CA_Net_HT', 'Marge_Nette_Magasin']  For example: Fetching Records with Numero_Transaction 'TID0000048821 cursor.execute("select * from Transactions where Point_de_Vente= 'PDV-id-00562") results = cursor.fetchall() print(results)  Database Update: Appending to "Products table"
	<pre>df2.to_sql('Products', connection, if_exists='append', index=False) connection.commit()  IntegrityError</pre>
	715 716 717 718 718 718 719 719 719 718 7224 7225 7226 7227 7228 7228 7228 738 749 740 750 750 750 750 760 760 760 760 760 760 760 760 760 76
]: [	<pre>~\anaconda3\lib\site-packages\pandas\io\sql.py in _execute_insert(self, conn, keys, data_iter)</pre>
	Tasks  1) Aggregating Total Margin for Signature Products over the Last 2 Months  # SQL Query sql_query_1 = """ SELECT p.SIGNATURE_PRODUCT, SUM(t.Marge_Nette_Magasin) as Total_Margin FROM products p JOIN Transactions t ON p.CODE_MODELE_COULEUR_ACTUEL = t.Modele_Couleur_Ref WHERE t.Date_Transaction >= date('now', '-2 months') GROUP BY p.SIGNATURE_PRODUCT;  # Execute the query cursor.execute(sql_query_1)
	<pre>#Fetch and display the results results = cursor.fetchall() for row in results:     print(row)  # Fetch and display the results #results = cursor.fetchall() #print(results)  (0, 5.483514000027069e+18) (1, 2.2720770000532713e+18)  2) Revenue Analysis: Daily Totals by "Transaction Date" (Chiffre d'Affaires_Net_Toutes Taxes Comprises = CA_Net_TTC)  # SQL Query sql_query_2 = """ SELECT t.Date_Transaction, SUM(t.CA_Net_TTC) as Daily_Revenue</pre>
	FROM Transactions t GROUP BY t.Date_Transaction ORDER BY t.Date_Transaction DESC; """  # Execute the query cursor.execute(sql_query_2)  # Fetch results results = cursor.fetchall()  # Fetch column names from the cursor description columns = [desc[0] for desc in cursor.description]  # Convert results to a DataFrame df_results = pd.DataFrame(results, columns=columns)  # Display the DataFrame
	Date_Transaction   Daily_Revenue     0
(	66 02/02/2022 2.836700e+04 67 01/02/2022 1.903800e+04 68 rows × 2 columns  3) Top 10 "Modele_Couleur_Ref" by Units Sold  # SQL Query sql_query3 = """ SELECT t.Modele_Couleur_Ref, SUM(t.Quantite_Vendue) as Total_Units_Sold FROM Transactions t GROUP BY t.Modele_Couleur_Ref ORDER BY Total_Units_Sold DESC LIMIT 10; """
	<pre># Execute the query cursor.execute(sql_query_3)  # Fetch results results = cursor.fetchall()  # Fetch column names from the cursor description columns = [desc[0] for desc in cursor.description]  # Convert results to a DataFrame df_results = pd.DataFrame(results, columns=columns)  # Display the DataFrame display(df_results)</pre> Modele_Couleur_Ref Total_Units_Sold
	o         MCR59750         70889           i         MCR19832         46018           z         MCR59751         42294           3         MCR94923         38475           4         MCR1987         37563           5         MCR55929         33611           6         MCR53294         30799           7         MCR94987         27531           8         MCR64986         26866           9         MCR66347         26809
	<pre>4) Number of Transactions per store  sql_query_4 = """ SELECT t.Point_de_Vente, COUNT(DISTINCT t.Numero_Transaction) as Total_Transactions FROM Transactions t GROUP BY t.Point_de_Vente ORDER BY Total_Transactions DESC; """  # Execute the query cursor.execute(sql_query_4)  # Fetch results results = cursor.fetchall()  # Fetch column names from the cursor description columns = [desc[0] for desc in cursor.description]</pre>
	# CONVERT results = pd. DataFrame   df_results = pd. DataFrame   columns   c
	df_results = pd.DataFrame(results, columns=columns)           # Display the DataFrame display(df_results)           Point_de_Vente Total_Transactions           0 PDV-id-0063         2906           1 PDV-id-00450         2814           2 PDV-id-00119         2640           3 PDV-id-00363         2609           4 PDV-id-00157         2606