Data Analytics Testing

Data Quality Checks for Data Warehouse/ETL

ETL testing refers to the process of validating, verifying, and qualifying data while preventing duplicate records and data loss.

In [1]:
```
from IPython import display
```

In [3]:
```
display.Image('/content/drive/MyDrive/Data_Engineering/SQL_ETL-600x428.png', width = 1400, height = 500)
```

Out[3]:

In [ ]:
```
#https://www.youtube.com/watch?v=3uWonpgXR1s
import sqlite3

# Connect to the database (or create it if it doesn't exist)
conn = sqlite3.connect('ETL_Testing.db')
```

```
# Create a cursor object to execute SQL commands
c = conn.cursor()
```

In [ ]:
```
!pip install csv
import pandas as pd
from pandas import DataFrame
from typing import List
from datetime import datetime
import csv
from google.colab import files
from google.colab import drive
!pip install openpyxl
!install urllib
import urllib
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
ERROR: Could not find a version that satisfies the requirement csv (from versions: none)
ERROR: No matching distribution found for csv
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: openpyxl in /usr/local/lib/python3.8/dist-packages (3.0.10)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.8/dist-packages (from openpyxl) (1.1.0)
install: missing destination file operand after 'urllib'
Try 'install --help' for more information.
```

In [ ]:
```
#import excel to colab with pandas
#You can use the Pandas library to import an Excel file into Google Colab.

#First, you need to install the pandas and openpyxl libraries:
```

In [ ]:
```
!pip install xlrd
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: xlrd in /usr/local/lib/python3.8/dist-packages (1.2.0)
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
nt=True).
```

In [ ]:
```
########## From csv to pandas dataframe in colab #############
```

In [ ]:
```
path0 = '/content/drive/MyDrive/Data_Engineering/ETL_Testing_Data QM/Copy of Customers.csv.xls - Copy of Customers.csv.x
import pandas as pd
#pd.read_csv(url,error_bad_lines=False)
```

```
Customers_pd = pd.read_csv(path0)
Customers_pd
```

Out[ ]:

	Customer Name	Province	Region	Customer Segment
0	Muhammed MacIntyre	Nunavut	Nunavut	Small Business
1	Barry French	Nunavut	Nunavut	Consumer
2	Clay Rozendal	Nunavut	Nunavut	Corporate
3	Carlos Soltero	Nunavut	Nunavut	Consumer
4	Carl Jackson	Nunavut	Nunavut	Corporate
...	...	...	...	...
790	Cyma Kinney	Alberta	West	Small Business
791	Shui Tom	Alberta	West	Home Office
792	Victoria Brennan	Alberta	West	Small Business
793	Adrian Shami	Alberta	West	Consumer
794	Harry Greene	Alberta	West	Corporate

795 rows × 4 columns

In [ ]:
```
Customers_pd.to_sql('Customers_Table', conn,if_exists="replace")
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/generic.py:2872: UserWarning: The spaces in these column names will
not be changed. In pandas versions < 0.14, spaces were converted to underscores.
  sql.to_sql(
```

In [ ]:
```
#pd.read_sql('''SELECT * FROM Customers_Table;''',conn)
```

In [ ]:
```
path1 = '/content/drive/MyDrive/Data_Engineering/ETL_Testing_Data QM/Orders.csv.xls - Orders.csv.xls.csv'
import pandas as pd
#pd.read_csv(url,error_bad_lines=False)
Orders_pd = pd.read_csv(path1)
Orders_pd
```

Out[ ]:

	Row ID	Order ID	Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Product Name	Ship Date
0	1	3	2010-10-13	Low	6	261.5400	0.04	Regular Air	-213.25	38.94	35.00	Muhammed MacIntyre	Eldon Base for stackable storage shelf. platinum	2010-10-20
1	49	293	2012-10-01	High	49	10123.0200	0.07	Delivery Truck	457.81	208.16	68.02	Barry French	1.7 Cubic Foot Compact "Cube" Office Refrigera...	2012-10-02
2	50	293	2012-10-01	High	27	244.5700	0.01	Regular Air	46.71	8.69	2.99	Barry French	Cardinal Slant-D® Ring Binder. Heavy Gauge Vinyl	2012-10-03
3	80	483	2011-07-10	High	30	4965.7595	0.08	Regular Air	1198.97	195.99	3.99	Clay Rozendal	R380	2011-07-12
4	85	515	2010-08-28	Not Specified	19	394.2700	0.08	Regular Air	30.94	21.78	5.94	Carlos Soltero	Holmes HEPA Air Purifier	2010-08-30
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8394	7765	55558	2010-08-09	Medium	8	1294.0400	0.05	Delivery Truck	-323.18	150.98	66.27	Mick Brown	Bush Mission Pointe Library	2010-08-09
8395	7766	55558	2010-08-09	Medium	23	392.5700	0.04	Regular Air	22.25	17.07	8.13	Mick Brown	Recycled Interoffice Envelopes with Re-Use-A-S...	2010-08-11
8396	7906	56550	2011-04-08	Not Specified	37	823.7800	0.03	Express Air	343.05	22.23	5.08	Frank Hawley	Executive Impressions 14"	2011-04-10

	Row ID	Order ID	Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Product Name	Ship Date
8397	7907	56550	2011-04-08	Not Specified	8	469.8375	0.00	Regular Air	-159.24	65.99	8.99	Frank Hawley	Talkabout T8367	2011-04-09
8398	7914	56581	2009-02-08	High	20	2026.0100	0.10	Express Air	580.43	105.98	13.99	Grant Donatelli	Tenex 46" x 60" Computer Anti-Static Chairmat....	2009-02-11

8399 rows × 14 columns

In [ ]:
```
for col in Orders_pd:
    print(col)
```

```
Row ID
Order ID
Order Date
Order Priority
Order Quantity
Sales
Discount
Ship Mode
Profit
Unit Price
Shipping Cost
Customer Name
Product Name
Ship Date
```

In [ ]:
```
path2 = '/content/drive/MyDrive/Data_Engineering/ETL_Testing_Data QM/Products.csv.xls - Products.csv.xls.csv'
import pandas as pd
#pd.read_csv(url,error_bad_lines=False)
Product_pd = pd.read_csv(path2)
#Product_pd
```

In [ ]:
```
path3 = '/content/drive/MyDrive/Data_Engineering/ETL_Testing_Data QM/RegionManagers.csv.xls - RegionManagers.csv.xls.csv
import pandas as pd
#pd.read_csv(url,error_bad_lines=False)
Regions_pd = pd.read_csv(path3)
Regions_pd
```

Out[ ]:

	Region	Manager
0	Atlantic	Chris
1	Northwest Territories	Erin
2	Nunavut	Sam
3	Ontario	William
4	West	Pat
5	Prarie	Pat
6	Quebec	Pat
7	Yukon	Pat

In [ ]:
```
grouped_data = Regions_pd .groupby("Region").agg({"Manager":"unique"})
```

In [ ]:
```
grouped_data
```

Out[ ]:

	Manager
Region	
Atlantic	[Chris]
Northwest Territories	[Erin]
Nunavut	[Sam]
Ontario	[William]
Prarie	[Pat]
Quebec	[Pat]
West	[Pat]
Yukon	[Pat]

In [ ]:
```
path4 = '/content/drive/MyDrive/Data_Engineering/ETL_Testing_Data QM/Returns.csv.xls - Returns.csv.xls.csv'
import pandas as pd
#pd.read_csv(url,error_bad_lines=False)
```

```
Returns_pd = pd.read_csv(path4)
Returns_pd
```

Out[ ]:

	Order ID	Status
0	1	Delivered
1	49	Delivered
2	50	Delivered
3	80	Delivered
4	85	Delivered
...	...	...
8394	7765	Delivered
8395	7766	Delivered
8396	7906	Delivered
8397	7907	Delivered
8398	7914	Delivered

8399 rows × 2 columns

In [ ]:
```
#### Using the to_sql() function in Pandas to load data from a DataFrame into an SQLite database.
```

In [ ]:
```
Customers_pd.to_sql('Customers_Table', conn, if_exists='replace')
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/generic.py:2872: UserWarning: The spaces in these column names will
not be changed. In pandas versions < 0.14, spaces were converted to underscores.
  sql.to_sql(
```

In [ ]:
```
Orders_pd.to_sql('Orders_Table', conn, if_exists='replace')
```

In [ ]:
```
Product_pd.to_sql('Product_Table', conn, if_exists='replace')
```

In [ ]:
```
Regions_pd.to_sql('Regions_Table', conn, if_exists='replace')
```

In [ ]:
```
Returns_pd.to_sql('Returns_Tabe', conn, if_exists='replace')
```

In [ ]:
```
# Connected to a SQLite database and retrieve a list of all the tables in the database.
```

In [ ]:
```
cursor = conn.execute("SELECT name FROM sqlite_master WHERE type='table';")
```

In [ ]:
```
#Used the fetchall() method to retrieve the result of the query and print the tables' names.
```

In [ ]:
```
tables = cursor.fetchall()
for table in tables:
    print(table[0])
```

```
Customers_Table
Orders_Table
Product_Table
Regions_Table
Returns_Tabe
```

In [ ]:
```
### In the context of data quality, data requirements refer to the specific characteristics
##  that data must possess in order to be considered "high quality." These characteristics may include accuracy,
### Completeness, Consistency, Timeliness, and Relevance.
```

In [ ]:
```
### Data completeness refers to the degree to which data is complete and contains all of the necessary information.
```

In [ ]:
```
### Data uniqueness refers to the degree to which data is unique and does not contain duplicate values.
```

In [ ]:
```
### Data validity refers to the degree to which data is accurate and conforms to a predefined set of rules or constrain
```

In [ ]:
```
### Data consistency refers to the degree to which data is consistent and free of conflicting or contradictory informat
```

In [ ]:
```
### Data integrity refers to the degree to which data is accurate, consistent, and reliable.
```

In [ ]:
```
### Data profiling is the process of analyzing and examining data in order to understand its characteristics and proper
```

In [ ]:
```
### A view in SQLite is a virtual table that is based on the result of a SELECT statement.
```

In [ ]:
```
#A database view is a virtual table that does not store data itself, but retrieves data from one or more tables in a da

#A database table, on the other hand, is a physical table that stores data in a structured format. Tables are typically
```

In [ ]:
```
### Count of records with null values in Order Date and Order ID columns
```

In [ ]:
```
pd.read_sql('''SELECT * FROM Orders_Table
WHERE 'Order Date' IS NULL;''',conn)
```

Out[ ]:

index	Row ID	Order ID	Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Product Name	Ship Date

In [ ]:
```

```

In [ ]:
```
pd.read_sql('''SELECT * FROM Orders_Table
WHERE 'Order ID' IS NULL;''',conn)
```

Out[ ]:

index	Row ID	Order ID	Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name	Product Name	Ship Date

In [ ]:
```

```