**ETL Pipeline Documentation**

**Project Overview**

This project focuses on designing and implementing an ETL (Extract, Transform, Load) pipeline for Gartner's IT Services Market Share data from 2020 to 2023. The goal is to centralize access, analysis, and visualization of the data, exploring market, vendor, and segment-level trends by year, geography, and verticals.

**Problem Statement**

Gartner has market share data from 2018 to 2020, but no centralized way to access, analyze, or view the information. The data's definitions for market, country, and vendor revenue have changed over time. This project addresses these challenges by building an efficient ETL pipeline for data integration and analysis.

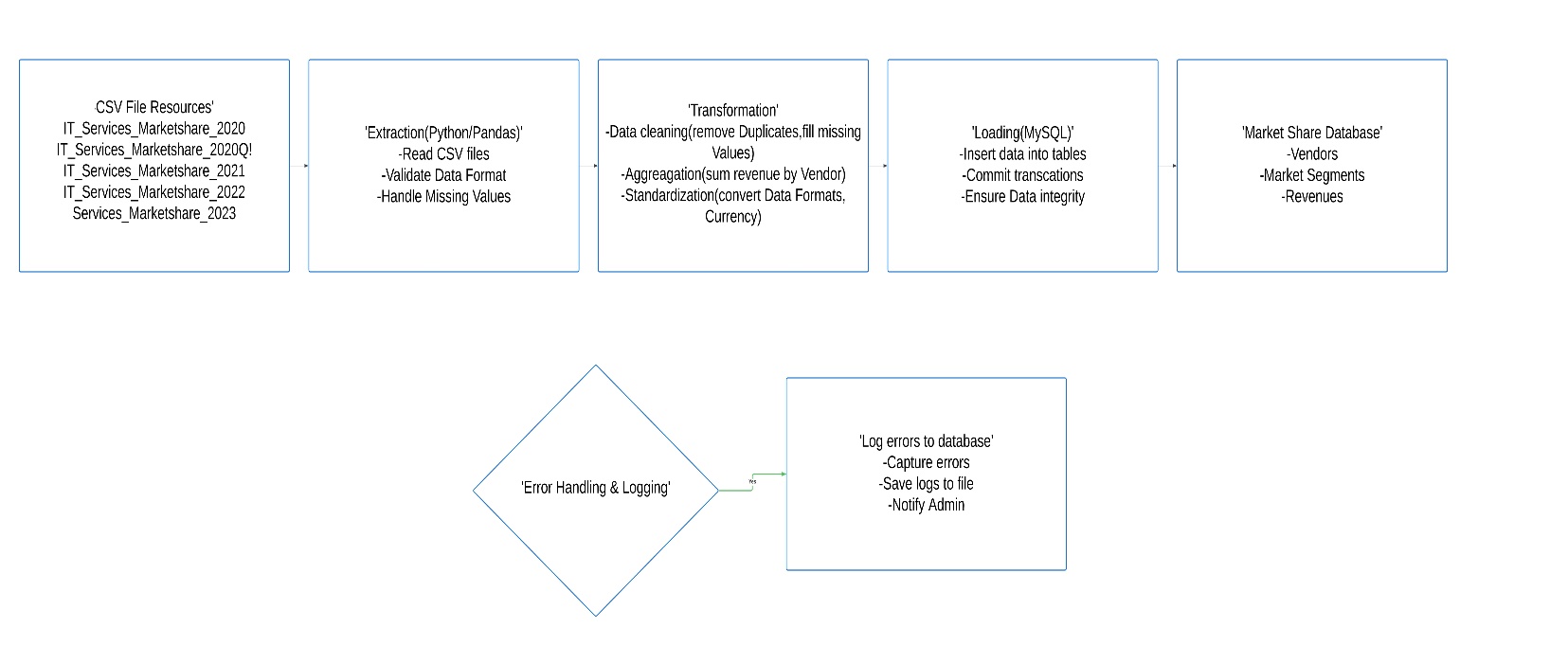
**Design Architecture**

The ETL pipeline consists of the following segments:

1. **CSV File Resources**:
   * IT\_Services\_Marketshare\_2020
   * IT\_Services\_Marketshare\_2020Q1
   * IT\_Services\_Marketshare\_2021
   * IT\_Services\_Marketshare\_2022
   * IT\_Services\_Marketshare\_2023
2. **Extraction (Python/Pandas)**:
   * Reading CSV files.
   * Validating data formats.
   * Handling missing values.
3. **Transformation**:
   * Data cleaning: Remove duplicates and fill missing values.
   * Aggregation: Sum revenue by vendor.
   * Standardization: Convert data formats and currency.
4. **Loading (MySQL)**:
   * Inserting data into tables.
   * Committing transactions.
   * Ensuring data integrity.
5. **Market Share Database**:
   * Tables for vendors, market segments, and revenues.

**Data Flow Diagram**

The data flow diagram includes:



**Data Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source Column** | **Target Column** | **Data Type** | **Transformation Needed** |
| Year | Year | INT | Convert to integer |
| Region | Region | VARCHAR(100) | Standardize naming |
| Country | Country | VARCHAR(50) | Convert to uppercase |
| Vendor | Vendor | VARCHAR(255) | Trim spaces, capitalize words |
| Service 1 | Service\_1 | VARCHAR(100) | Remove special characters |
| Service 2 | Service\_2 | VARCHAR(100) | Remove special characters |
| Service 3 | Service\_3 | VARCHAR(100) | Remove special characters |
| Vertical | Vertical | VARCHAR(100) | Standardize naming |
| Ticker | Ticker | VARCHAR(50) | Convert to uppercase |
| HQ Country | HQ\_Country | VARCHAR(50) | Convert to uppercase |
| VendorRevenue - USD | VendorRevenue\_USD | DECIMAL(15,2) | Remove $ and commas |
| ConstantCurrency Revenue - USD | ConstantCurrency\_Revenue\_USD | DECIMAL(15,2) | Remove $ and commas |

**Implementation**

**MySQL Database**

The database and table were created using the following SQL:

create database market\_data

use market\_data;

create table market\_data (

Year INT,

Super\_Region VARCHAR(255),

Region VARCHAR(255),

Country VARCHAR(255),

Vendor VARCHAR(255),

Service\_1 VARCHAR(255),

Service\_2 VARCHAR(255),

Service\_3 VARCHAR(255),

Vertical VARCHAR(255),

Ticker VARCHAR(255),

HQ\_Country VARCHAR(255),

VendorRevenue\_USD DECIMAL(20, 7),

ConstantCurrency\_Revenue\_USD DECIMAL(20, 7)

);

select\* from market\_data;

select count(\*) from market\_data;

# Data Validation

SELECT COUNT(\*)

FROM market\_data

WHERE Year IS NULL or Super\_Region is null or Region is null or Vendor is null or Service\_1 is null or Service\_2 is null or Service\_3 is null or Vertical is null or Ticker is null or HQ\_Country is null or VendorRevenue\_USD is null or ConstantCurrency\_Revenue\_USD is null;

SELECT Year, Vendor, Country, Service\_1, Service\_2, Service\_3, Vertical, Ticker, HQ\_Country, VendorRevenue\_USD, ConstantCurrency\_Revenue\_USD, COUNT(\*) as Duplicate\_Count

FROM market\_data

GROUP BY Year, Vendor, Country, Service\_1, Service\_2, Service\_3, Vertical,Ticker, HQ\_Country, VendorRevenue\_USD, ConstantCurrency\_Revenue\_USD

HAVING COUNT(\*) > 1

limit 1000;

# Performance Optimization

EXPLAIN

SELECT Year, Vendor, Country, Service\_1, Service\_2, Service\_3, Vertical, Ticker, HQ\_Country,

VendorRevenue\_USD, ConstantCurrency\_Revenue\_USD, COUNT(\*) AS Duplicate\_Count

FROM market\_data

GROUP BY Year, Vendor, Country, Service\_1, Service\_2, Service\_3, Vertical, Ticker, HQ\_Country,

VendorRevenue\_USD, ConstantCurrency\_Revenue\_USD

HAVING COUNT(\*) > 1;

CREATE INDEX idx\_year ON market\_data (Year);

CREATE INDEX idx\_vendor\_country ON market\_data (Vendor, Country);

CREATE INDEX idx\_revenue ON market\_data (VendorRevenue\_USD);

CREATE INDEX idx\_currencyrevenue ON market\_data (ConstantCurrency\_Revenue\_USD);

SHOW INDEXES FROM market\_data;

SHOW TABLE STATUS LIKE 'mmarket\_data';

SHOW VARIABLES LIKE 'max\_allowed\_packet';

SHOW VARIABLES LIKE 'wait\_timeout';

SHOW VARIABLES LIKE 'innodb\_buffer\_pool\_size';

OPTIMIZE TABLE market\_data;

ANALYZE TABLE market\_data;

# key bottlenecks

SELECT Year, Vendor, SUM(VendorRevenue\_USD)

FROM market\_data

GROUP BY Year, Vendor

ORDER BY Year;

SELECT

Vendor,

AVG(VendorRevenue\_USD) AS avg\_revenue,

MAX(VendorRevenue\_USD) AS max\_revenue,

MIN(VendorRevenue\_USD) AS min\_revenue

FROM market\_data

GROUP BY Vendor

ORDER BY max\_revenue DESC

LIMIT 100;

**Python ETL Process**

The ETL process was implemented in Python, including the following steps:

1. **Uploading CSV Files**:
   * Used Pandas to read CSV files.
   * Processed data in chunks to handle large datasets.
2. **Data Transformation**:
   * Standardized column names.
   * Renamed specific columns to match the SQL table schema.
   * Replaced missing values with "Unknown."
   * Converted currency fields to numeric values.
3. **Data Loading**:
   * Inserted data into the MySQL database using the mysql.connector library.
4. **Error Handling and Logging**:
   * Logged errors to a log file (ETL\_Pipeline.log).

**Performance Optimization**

The following measures were implemented:

* Created indexes for frequently queried columns (e.g., Year, Vendor, Country, Revenue).
* Optimized large dataset queries with EXPLAIN and indexing.
* Tuned MySQL server settings (e.g., max\_allowed\_packet, wait\_timeout).

**Python Code**

import pandas as pd

import mysql.connector

import logging

# For logging configuration

logging.basicConfig(

    filename= 'ETL\_Pipeline.log',

    level=logging.INFO,

    format='%(asctime)s - %(levelname)s - %(message)s'

)

# Uploading of CSV files

CSV\_FILES = [

    r"D:\ETL\_Project\IT\_Services\_Marketshare\_2019.csv",

    r"D:\ETL\_Project\IT\_Services\_Marketshare\_2020.csv",

    r"D:\ETL\_Project\IT\_Services\_Marketshare\_2021.csv",

    r"D:\ETL\_Project\IT\_Services\_Marketshare\_2022.csv",

    r"D:\ETL\_Project\IT\_Services\_Marketshare\_2023.csv"

]

# MySQL connection details

mydb = mysql.connector.connect(

    host="localhost",

    user="root",

    password="root",

    database="market\_data"

)

DB\_TABLE = "market\_data"

logging.info('Connected to MySQL successfully')

#This function is to retrieve table columns from the database

def get\_table\_columns(connection, table\_name):

    cursor = connection.cursor()

    cursor.execute(f"DESCRIBE {table\_name}")

    columns = [row[0] for row in cursor.fetchall()]

    cursor.close()

    return columns

# Function to process each chunk of data and insert it into MySQL

def process\_chunk(chunk, table\_name, connection, table\_columns):

    try:

        # Standardization of column names

        chunk.columns = [col.strip().replace(" - ", "\_").replace(" ", "\_").replace("-", "\_") for col in chunk.columns]

        # Rename of specific columns to match SQL table schema

        chunk.rename(columns={

            'Super\_Region': 'Super\_Region',

            'HQ\_Country': 'HQ\_Country',

            'VendorRevenue\_USD': 'VendorRevenue\_USD',

            'ConstantCurrency\_Revenue\_USD': 'ConstantCurrency\_Revenue\_USD',

        }, inplace=True)

        # For removal of 'Yr' from Year col

        if 'Year' in chunk.columns:

            chunk['Year'] = chunk['Year'].astype(str).str.replace(' YR', '', regex=False).astype(int)

        # Replacing missing values

        chunk.fillna("Unknown", inplace=True)

        # Transformation of currency fields to numeric values

        currency\_columns = ["VendorRevenue\_USD", "ConstantCurrency\_Revenue\_USD"]

        for col in currency\_columns:

            if col in chunk.columns:

                chunk[col] = chunk[col].replace({r'\$': '', ',': '', ' ': ''}, regex=True)

                chunk[col] = pd.to\_numeric(chunk[col], errors='coerce').fillna(0)

        # For the alignmwnt of columns with the database schema

        chunk = chunk[[col for col in chunk.columns if col in table\_columns]]

        # Inserting the data into MySQL

        cursor = connection.cursor()

        cols = ", ".join(chunk.columns)

        placeholders = ", ".join(["%s"] \* len(chunk.columns))

        insert\_query = f"INSERT INTO {table\_name} ({cols}) VALUES ({placeholders})"

        # Inserting all rows in the chunk

        cursor.executemany(insert\_query, chunk.values.tolist())

        connection.commit()

        cursor.close()

        logging.info(f"{len(chunk)} rows processed and inserted successfully.")

    except Exception as e:

        logging.error(f"Error processing chunk: {e}")

# Get the database table schema to ensure column alignment

table\_columns = get\_table\_columns(mydb, DB\_TABLE)

# Process each CSV file in chunks

for file in CSV\_FILES:

    try:

        logging.info(f"Processing file: {file}")

        chunksize = 100000  # Adjust chunk size based on memory capacity

        for chunk in pd.read\_csv(file, chunksize=chunksize):

            process\_chunk(chunk, DB\_TABLE, mydb, table\_columns)

        logging.info(f"Finished processing file: {file}")

    except Exception as e:

        logging.error(f"Error processing file {file}: {e}")

# Close the database connection

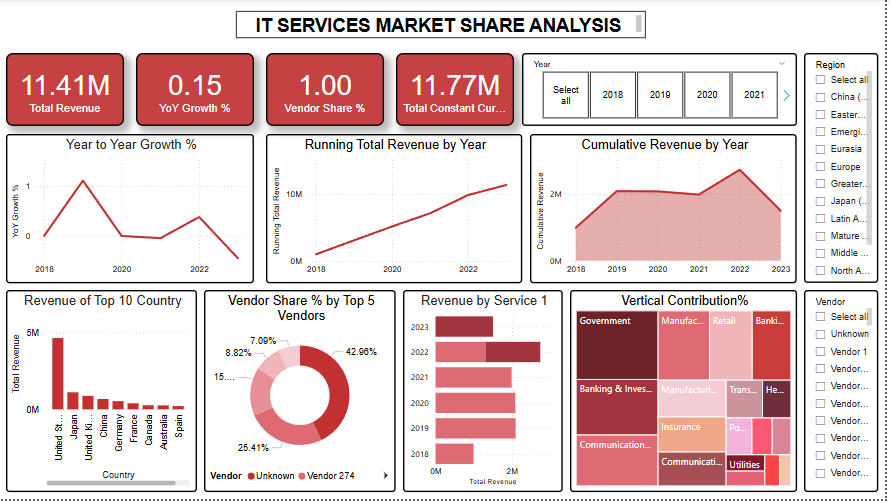
mydb.close()

logging.info("Database connection closed.")

**Visualization**

The data was visualized in Power BI with the following features:

1. **Measures**:
   * Cumulative Revenue
   * Previous Year Revenue
   * Revenue Growth Rate
   * Running Total Revenue
   * Selected Vendor Revenue
   * Total Revenue by Region
2. **Visualizations**:
   * **KPI Cards**: Total Revenue, YoY Growth %, Vendor Share %, Total Constant Currency Revenue.
   * **Line Chart**: YoY Growth % by Year.
   * **Area Chart**: Cumulative Revenue by Year.
   * **Bar Chart**: Revenue of Top 10 Countries.
   * **Pie Chart**: Vendor Share % by Top 5 Vendors.
   * **Treemap**: Vertical Contribution % by Verticals.
   * **Slicers**: Year, Region, Vendor.



**Documentation and Communication**

**Setup Instructions**

1. Install Python, Pandas, MySQL, and Power BI.
2. Set up the MySQL database and table schema.
3. Configure the ETL pipeline Python script.
4. Load data into the MySQL database.
5. Connect Power BI to the database for visualization.

**Assumptions**

* All CSV files follow a consistent schema.
* Missing values are replaced with default values (e.g., "Unknown").
* Currency fields are properly formatted and convertible to numeric values.

**Conclusion**

The ETL pipeline successfully integrates IT Services Market Share data from 2018 to 2023, providing centralized access and powerful visualization capabilities. The project showcases a comprehensive approach to data engineering, performance optimization, error handling, and reporting.

**Contact Information:**

**Name:** Lopita Mishra  
**Email:** lopitamishra2001@gmail.com  
**Phone:** +91 7848910741