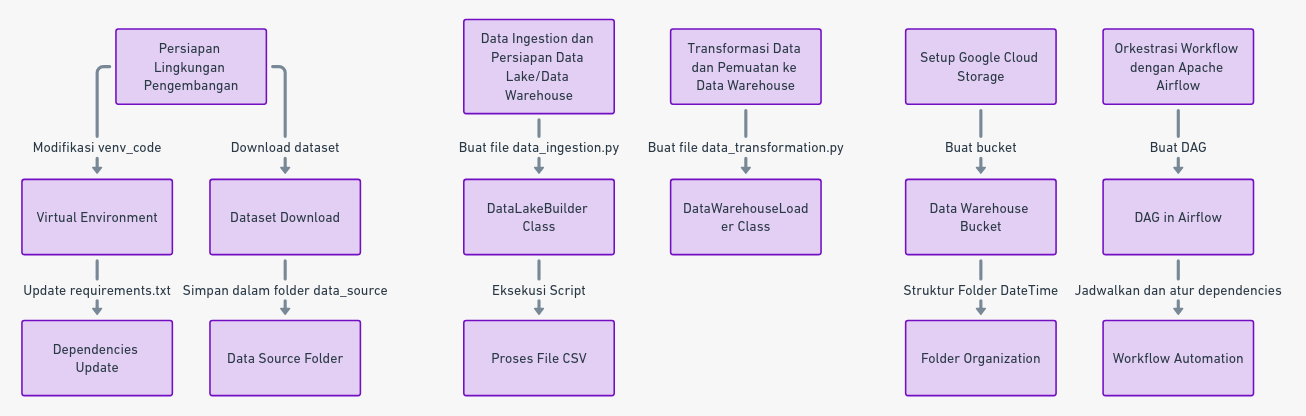
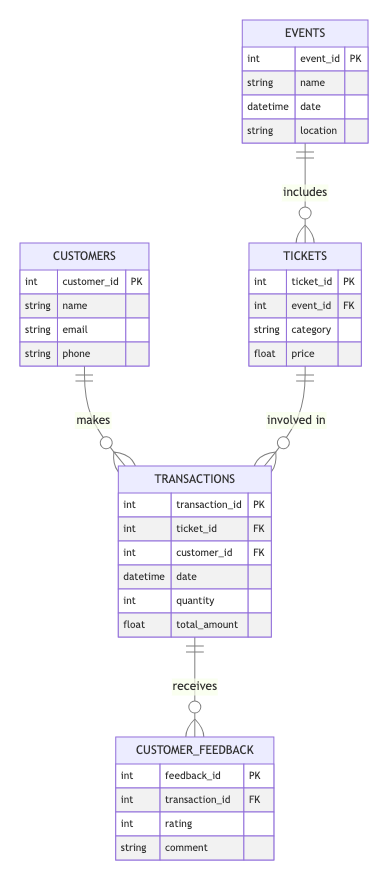
📝 Task



1. **Persiapan Lingkungan Pengembangan**

* Gunakan virtual environment **venv\_code** yang telah ada. Modifikasi environment ini untuk memasukkan dependencies baru yang diperlukan, seperti library pandas, airflow, dan library lain yang diperlukan.
* Update **requirements.txt** untuk mencerminkan perubahan tersebut.
* Download [dataset](https://www.google.com/url?q=https://gist.github.com/yudhaislamisulistya/fb643e0e4a740ebe8df959899e6eb54f&sa=D&source=editors&ust=1715094493853445&usg=AOvVaw1qc38TTQMavAwq9emOZFpk) dan kemudian simpan kedalam folder **data\_source**
* Struktur file csv



1. **Data Ingestion dan Persiapan Data Lake / Data Werehouse**

* Penyiapan Skrip Python
* Buat file Python dengan nama **data\_ingestion.py**.
* Dalam file ini, buat kelas dengan nama **DataLakeBuilder**.
* Kelas DataLakeBuilder
* Kelas ini harus memiliki metode untuk membaca data dari file CSV, menangani missing values dan outliers, dan menyimpan data ke format yang sesuai untuk Data Lake.
* **read\_csv\_data(self, file\_path)**: Metode untuk membaca data dari file CSV.
* **handle\_missing\_values(self, df)**: Metode untuk menangani missing values dalam DataFrame.
* **handle\_outliers(self, df, column)**: Metode untuk menangani outliers dalam kolom tertentu dari DataFrame.
* **save\_to\_parquet(self, df, file\_name)**: Metode untuk menyimpan DataFrame ke file Parquet.
* Metode **read\_csv\_data**
* Gunakan pandas untuk membaca file CSV.
* Contoh: **pd.read\_csv(file\_path)** di mana **file\_path** adalah path ke file CSV.
* Metode **handle\_missing\_values**
* Terapkan strategi untuk menangani missing values, seperti pengisian nilai (imputation) atau penghapusan baris/kolom.
* Contoh: Menggunakan **df.fillna()** atau **df.dropna()** dari pandas.
* Metode**handle\_outliers**
* Gunakan teknik statistik untuk mendeteksi dan menangani outliers.
* Contoh: IQR (Interquartile Range) untuk mengidentifikasi outliers dan kemudian menggantinya atau menghapusnya.
* Metode**save\_to\_parquet**
* Simpan DataFrame yang telah diolah ke format Parquet, yang efisien untuk Data Lake.
* Contoh: df.to\_parquet(file\_name).
* Eksekusi Script
* Buat instance dari **DataLakeBuilder** dan panggil metode-metodenya untuk setiap file CSV (**events.csv,** **customers.csv**, **tickets.csv**, **transactions.csv**, dan **customer\_feedback.csv**).
* Pastikan semua file terbaca dan diolah dengan benar, lalu simpan dalam format Parquet.
* Validasi Data
* Setelah semua data disimpan dalam format Parquet, buat metode untuk memvalidasi atau menampilkan ringkasan data.
* Contoh: Metode **validate\_data(self, file\_path)** yang membaca file Parquet dan menampilkan ringkasan statistik atau beberapa baris awal data.

1. **Transformasi Data dan Pemuatan ke Data Warehouse**

* Penyiapan Skrip Python untuk Transformasi Data
* Buat file Python baru dengan nama **data\_transformation.py**.
* Dalam file ini, buat kelas dengan nama **DataWarehouseLoader**.
* Kelas DataWarehouseLoader
* Kelas ini harus memiliki metode untuk membaca data dari Data Lake, melakukan transformasi data, dan menyimpan data yang telah ditransformasi ke Data Warehouse.
* **load\_data(self, file\_path)**: Metode untuk membaca data dari Data Lake (format Parquet).
* **transform\_data(self, df)**: Metode umum untuk melakukan transformasi data.
* **save\_to\_warehouse(self, df, table\_name)**: Metode untuk menyimpan data ke Data Warehouse.
* Metode **load\_data**
* Gunakan pandas untuk membaca file Parquet dari Data Lake.
* Load data dari file Parquet untuk tabel **events**, **customers**, **tickets**, **transactions**, dan **customer\_feedback**.
* Contoh: pd.read\_parquet(file\_path).
* Metode **Transform Data**
* Menggabungkan Data untuk Analisis
* Gabungkan tabel **transactions** dengan **tickets** berdasarkan **ticket\_id**, dan kemudian dengan **events** berdasarkan **event\_id**.
* Menghitung Jumlah Tiket yang Terjual per Acara
* Gunakan **groupby** pada kolom **event\_id** atau **name** dari tabel **events**.
* Hitung jumlah **tiket** yang terjual dengan mengagregasi kolom **quantity** menggunakan fungsi **sum**.
* Menghitung Total Pendapatan dari Setiap Acara
* Dari DataFrame yang digabungkan sebelumnya, gunakan **groupby** pada kolom **event\_id** atau **name**.
* Hitung total pendapatan dengan mengagregasi **total\_amount** menggunakan fungsi **sum**.
* Analisis Feedback Pelanggan
* Gabungkan tabel **customer\_feedback** dengan **transactions** berdasarkan **transaction\_id** untuk mengaitkan **feedback** dengan transaksi.
* Analisis rating rata-rata per acara.
* Metode **Save to Werehouse**
* Penamaan File Hasil Transformasi
* Tickets\_sold\_per\_event.parquet
* Revenue\_per\_event.parquet
* Feedback\_analysis.parquet
* Setup Google Cloud Storage
* Pastikan Anda memiliki akses ke Google Cloud Storage.
* Buat bucket yang akan digunakan sebagai Data Warehouse, jika belum ada.
* Struktur Folder Berdasarkan DateTime
* Untuk mengorganisir data, gunakan struktur folder berdasarkan tanggal. Format yang umum adalah **YYYY/MM/DD** atau **YYYY-MM-DD**.
* Contoh: Untuk data yang diproses pada tanggal **20 November 2023**, folder bisa bernama **2023/11/20** atau **2023-11-20**.

1. **Orkestrasi Workflow dengan Apache Airflow**

* Buatlah DAG (Directed Acyclic Graph) di Apache Airflow untuk mengotomatisasi proses pengambilan data, transformasi, dan pemuatan ke Data Warehouse.
* DAG harus mencakup task untuk data ingestion, data transformation, dan loading data ke warehouse.
* Jelaskan bagaimana Anda akan menjadwalkan DAG ini dan mengatur dependencies antar tasks.

📝 Note

1. Tugas dikumpulkan melalui Github.
2. Dokumentasikan setiap langkah-langkah diatas