**Business Requirements Document (BRD)**

**Project Title: Zoom Car Data Processing Pipeline**

**Objective**

The goal of this project is to design and implement a data processing pipeline for processing car booking and customer data. The pipeline will automate the extraction, transformation, and loading (ETL) of daily car booking and customer information, and facilitate merging of this data for downstream analytics.

**1. Background**

Zoom Car, a car rental service, maintains two datasets:

* **Car Booking Data**: Details about car bookings including the booking ID, customer ID, car ID, booking dates, start and end times, and status.
* **Customer Data**: Information about customers, including customer ID, name, email, phone number, signup date, and account status.

These datasets are stored in a Google Cloud Storage (GCP) bucket and updated daily. Our objective is to process and clean this data, load it into Delta tables, and perform transformation and merging operations to ensure accuracy and consistency.

**2. Scope**

The project will involve the following tasks:

1. **Data Preparation**: Generation of mock sample data for both car bookings and customer records.
2. **PySpark Notebooks**: Creation of two separate notebooks for processing the car booking and customer datasets, along with a third notebook for merging data.
3. **Transformations**: Apply transformations on data as per the business logic.
4. **Automation**: Develop a Databricks job to automate the data processing pipeline.
5. **Deliverables**: The project will deliver sample datasets, PySpark notebooks for bookings, customers, and data merging, as well as a configuration JSON for the Databricks workflow.

**3. Functional Requirements**

**3.1 Data Files and Sample Data**

The system will handle two types of data files stored in the GCP bucket:

1. **Zoom Car Bookings Data** (zoom\_car\_bookings\_yyyymmdd.json):
   * Fields:
     + booking\_id: Unique ID for the booking (string)
     + customer\_id: Unique ID for the customer (string)
     + car\_id: Unique ID for the car (string)
     + booking\_date: Date of booking (YYYY-MM-DD)
     + start\_time: Booking start time (ISO 8601 format)
     + end\_time: Booking end time (ISO 8601 format)
     + total\_amount: Total amount for the booking (decimal)
     + status: Status of the booking (e.g., completed, cancelled, pending)

Sample JSON

[

    {

      "booking\_id": "B001",

      "customer\_id": "C001",

      "car\_id": "CAR123",

      "booking\_date": "2024-07-20",

      "start\_time": "2024-07-21T10:00:00Z",

      "end\_time": "2024-07-21T18:00:00Z",

      "total\_amount": 150.75,

      "status": "completed"

    },

    {

      "booking\_id": "B002",

      "customer\_id": "C002",

      "car\_id": "CAR456",

      "booking\_date": "2024-07-20",

      "start\_time": "2024-07-21T12:00:00Z",

      "end\_time": "2024-07-21T16:00:00Z",

      "total\_amount": 80.50,

      "status": "cancelled"

    }

  ]

1. **Zoom Car Customers Data** (zoom\_car\_customers\_yyyymmdd.json):

* Fields:
  + customer\_id: Unique ID for the customer (string)
  + name: Name of the customer (string)
  + email: Email address of the customer (string)
  + phone\_number: Phone number of the customer (string)
  + signup\_date: Date the customer signed up (YYYY-MM-DD)
  + status: Account status (e.g., active, inactive)
* [
* {
* "customer\_id": "C001",
* "name": "John Doe",
* "email": "john.doe@example.com",
* "phone\_number": "1234567890",
* "signup\_date": "2024-01-15",
* "status": "active"
* },
* {
* "customer\_id": "C002",
* "name": "Jane Smith",
* "email": "jane.smith@example.com",
* "phone\_number": "0987654321",
* "signup\_date": "2023-12-22",
* "status": "inactive"
* }
* ]

**3.2 PySpark Notebooks**

* **Notebook 1: Process Zoom Car Bookings**
  + **Input**: JSON file for the current date (zoom\_car\_bookings\_yyyymmdd.json).
  + **Processing**:
    - Remove records with null values in critical fields (booking\_id, customer\_id, car\_id, booking\_date).
    - Validate date formats.
    - Ensure status is one of the predefined statuses (completed, cancelled, pending).
    - Parse start\_time and end\_time into separate date and time columns.
    - Calculate the total duration of each booking.
  + **Output**: Cleaned data loaded into the staging\_bookings\_delta table.
* **Notebook 2: Process Zoom Car Customers**
  + **Input**: JSON file for the current date (zoom\_car\_customers\_yyyymmdd.json).
  + **Processing**:
    - Remove records with null values in critical fields (customer\_id, name, email).
    - Validate email formats.
    - Ensure status is one of the predefined statuses (active, inactive).
    - Normalize phone numbers to a standard format.
    - Calculate customer tenure from signup\_date.
  + **Output**: Cleaned data loaded into the staging\_customers\_delta table.
* **Notebook 3: Merge Data**
  + **Input**: Cleaned data from the staging tables.
  + **Processing**:
    - **Merge Conditions**:
      * **Update**: If booking\_id or customer\_id exists in the target table, update the existing records.
      * **Insert**: If booking\_id or customer\_id does not exist, insert new records.
      * **Delete**: If the status of a booking is cancelled, delete the record from the target table.
  + **Output**: Updated target Delta table.

**3.3 Parameterized Notebooks**

Both notebooks will accept the current date as a parameter and read the corresponding JSON files for the current date (zoom\_car\_bookings\_yyyymmdd.json and zoom\_car\_customers\_yyyymmdd.json).

**4. Non-Functional Requirements**

**4.1 Performance**

* The data processing pipeline should be optimized for daily execution.
* The merging operations should handle large datasets efficiently, ensuring minimal latency.

**4.2 Security**

* The GCP storage bucket should have proper access control and encryption mechanisms.
* The Databricks job should be secured with appropriate access rights.

**4.3 Automation**

* The pipeline will be automated using Databricks job workflows, triggered manually on a daily basis by passing the current date as a parameter.

**5. Deliverables**

* **Mock Data**: JSON files containing sample car booking and customer data.
* **PySpark Notebooks**:
  + Notebook 1: Process Zoom Car Bookings
  + Notebook 2: Process Zoom Car Customers
  + Notebook 3: Merge Data
* **JobFlow JSON**: Configuration for the Databricks job workflow that automates the ETL process.

**6. Assumptions**

* The provided sample data will be in the specified JSON format.
* Databricks workspace and required access permissions are already set up.
* All required dependencies (e.g., Delta Lake) are available in the Databricks environment.