# PRUEBA TECNICA

Analista de Procesos ETL

## Documento Explicativo y Argumentativo

Descripcion del enfoque, decisiones tecnicas
y justificacion de cada seccion desarrollada

Herramientas: Python (Jupyter), PostgreSQL (Docker), Excel

# TABLA DE CONTENIDO

# 1. MATRIZ DE CUMPLIMIENTO

Desarrollo completo de la prueba tecnica para Analista de Procesos ETL. Se abordan las cuatro secciones: Excel, Python, SQL y Logica ETL.

## Resumen por criterio

| Seccion / Criterio | Estado | Observacion |
|---|---|---|
| Excel - Cargue y anexar en Power Query | OK | Instrucciones documentadas + CSVs |
| Excel - Tabla Dinamica y Resultado | OK | Procedimiento detallado paso a paso |
| Python - Cargue e integracion | OK | 15,000 registros unificados correctamente |
| Python - Eliminacion de duplicados | OK | Deduplicacion por Col_1 con trazabilidad |
| Python - Validacion emails Col_8 | OK | Regex + clasificacion valido/invalido |
| Python - Validacion telefonos Col_11 | OK | Reglas colombianas celular/fijo |
| Python - Exportacion CSV limpio | OK | flights_unificado_limpio.csv generado |
| Python - Conexion a BD SQL | OK | pyodbc + SQLAlchemy documentados |
| SQL - Carga de datos | OK | COPY a FlightsBase y FlightsNew |
| SQL - Store Procedure upsert | OK | ON CONFLICT DO UPDATE completo |
| SQL - Resultado final sin duplicados | OK | 10,000 registros unicos verificados |
| Logica ETL - Caso de Uso | OK | Flujo completo con diagrama |
| Logica ETL - Buen uso de un ETL | OK | Partes A, B y C desarrolladas |

**Todos los criterios cumplidos satisfactoriamente**

## 2. SECCION 1 - EXCEL

### Que se hizo

- Se conectaron los dos CSV en Power Query, se corrigieron headers corruptos del archivo de 5000, se anexaron ambas consultas y se cargo el resultado.
- Se convirtio Col_10 de texto a numero decimal para permitir la suma.
- Se creo tabla dinamica con filas por Col_2, cuenta de Col_1 y suma de Col_10.
- Se identifico el valor mas duplicado de Col_1 en los 15,000 registros.

### Por que se hizo asi

- Headers corruptos: flights_5000v2.csv tiene Col_7 en posicion 6, Col_17 en posicion 16 y Col_13 en posicion 18. Se renombraron antes del Append para alinear columnas correctamente.
- Col_10 como numero decimal: los datos vienen mezclados entre texto y numero; sin esta conversion la tabla dinamica no puede calcular la suma.

## 3. SECCION 2 - PYTHON

### 3.1 Carga e integracion

- Se cargaron ambos CSV con pandas usando sep=";" y encoding="utf-8-sig" para manejar el BOM y el delimitador correcto.
- Se corrigieron los headers corruptos de flights_5000v2.csv con el parametro names=.
- Se unificaron con pd.concat() obteniendo 15,000 registros en flights_Union.

### 3.2 Eliminacion de duplicados

- Se uso drop_duplicates con subset=Col_1 y keep="first" para conservar la primera ocurrencia y priorizar los datos del archivo de 10,000.
- Se imprimieron conteos antes/despues para trazabilidad del proceso.

### 3.3 Validacion de emails - Col_8

- Se limpiaron ~200 espacios de padding con strip() antes de validar.
- Se aplico regex estandar de email para clasificar validos e invalidos. Valores como "NO TIENE" o "NA" se rechazan correctamente.

### 3.4 Validacion de telefonos - Col_11

- Se limpiaron caracteres +, -, espacios y se valido contra reglas colombianas: celular = 10 digitos empezando con 3, fijo = 10 digitos empezando con 601.

### 3.5 Exportacion

- Se exporto flights_Union completo con Col_8 y Col_11 depuradas como flights_unificado_limpio.csv (15,000 registros, encoding utf-8-sig).

### 3.6 Conexion a BD SQL

- Se documentaron las librerias pyodbc y SQLAlchemy con sus parametros de conexion a SQL Server (DRIVER, SERVER, DATABASE, UID, PWD) y ejemplos funcionales.

---

## 4. SECCION 3 - SQL

### Motor elegido: PostgreSQL 15 con Docker

- Se eligio PostgreSQL por soporte nativo de UPSERT (ON CONFLICT DO UPDATE), COPY para carga masiva de CSV y DISTINCT ON para deduplicacion. Docker permite levantar el ambiente con un solo comando sin instalacion local.

### 4.1 Carga de datos

- Se crearon tablas FlightsBase y FlightsNew (19 columnas VARCHAR) y se cargaron los CSV con COPY. Resultado: FlightsBase=5,000 y FlightsNew=10,000 registros.
- Columnas VARCHAR porque los datos tienen formatos mixtos (ej: "998E" en Col_4). Los archivos CSV se montan como volumen Docker en /data/ para acceso directo.

### 4.2 Store Procedure - sp_upsert_flights

- El SP deduplica ambas tablas (ctid para FlightsBase, DISTINCT ON para FlightsNew), agrega constraint UNIQUE en Col_1 y ejecuta INSERT ... ON CONFLICT DO UPDATE.
- Se eligio UPSERT nativo de PostgreSQL porque es atomico, eficiente e idempotente. Cada paso imprime RAISE NOTICE con conteos para auditoria del proceso.

### 4.3 Verificacion del resultado

- Cuatro queries de verificacion: COUNT vs COUNT DISTINCT, busqueda explicita de duplicados con HAVING, muestra de datos y validacion automatica OK/ERROR.
- Resultado esperado: FlightsBase con exactamente 10,000 registros unicos.

---

## 5. SECCION 4 - LOGICA ETL

### Parte A - Diseno del flujo

- Flujo de 6 etapas: Extraccion incremental con watermark, Validacion de schema y nulos con tabla de rechazo, Transformacion con deduplicacion y logica de negocio, Carga via staging + UPSERT transaccional, Manejo de errores con logging y alertas, y Monitoreo con dashboard y conteo por etapa.
- Se incluye diagrama ASCII del flujo completo desde las fuentes hasta el DWH.

### Parte B - Criterios de calidad

- Conteo de filas por etapa y checksums para detectar perdida de datos. Validacion con regex y rangos. Operaciones idempotentes (UPSERT) con tabla de control para evitar reprocesos. Deteccion de anomalias por

perfilado estadistico.

### Parte C - Continuidad y mejora

- Ante fallas: retry con backoff, Dead Letter Queue, rollback transaccional y checkpointing.
- Documentacion: diccionario de datos, linaje de datos y runbooks operativos.
- Mejoras futuras: migracion a streaming (Kafka), catalogo de datos y scoring de calidad.

# 6. GUIA DE EJECUCION PASO A PASO

A continuacion se detalla como ejecutar y validar cada seccion desde cero en cualquier computador Windows.

## Prerequisitos generales

- Python 3.8 o superior instalado (descargar de python.org)
- Visual Studio Code con extension "Jupyter" instalada
- Docker Desktop instalado (solo para Seccion SQL)
- Microsoft Excel 2016 o superior (solo para Seccion Excel)

Instalar las librerias de Python necesarias:

```
pip install pandas jupyter
```

## Ejecutar Seccion 1 - Excel

**Paso 1: Abrir Excel y conectar los archivos**

- Ir a Datos > Obtener datos > Desde texto/CSV
- Seleccionar flights_10000v2.csv, delimitador: Punto y coma (;)
- Clic en "Transformar datos" para abrir Power Query. Repetir para flights_5000v2.csv

**Paso 2: Corregir headers y anexar**

- En Power Query, renombrar Col_7->Col_6, Col_17->Col_16, Col_13->Col_18 en flights_5000v2
- Inicio > Anexar consultas > Como nueva > Seleccionar ambas tablas
- Cambiar tipo de Col_10 a "Numero decimal"

**Paso 3: Tabla dinamica**

- Cerrar y cargar > Tabla. Insertar > Tabla dinamica
- Filas: Col_2 | Valores: Cuenta de Col_1 + Suma de Col_10

## Ejecutar Seccion 2 - Python

**Paso 1: Abrir el notebook en VS Code**

- Archivo > Abrir carpeta > Seccion_2_Python
- Abrir ETL_Prueba_Tecnica.ipynb y seleccionar kernel de Python

**Paso 2: Ejecutar y verificar**

- Clic en "Run All" o ejecutar celda por celda con Shift+Enter
- Celda 1: Debe mostrar 15,000 registros en flights_Union
- Celda 2: Muestra antes/despues de deduplicar
- Celda 3: Conteo de emails validos/invalidos
- Celda 4: Conteo de telefonos aptos/no aptos
- Celda 5: Genera flights_unificado_limpio.csv

## Ejecutar Seccion 3 - SQL

**Paso 1: Levantar PostgreSQL con Docker**

- Abrir terminal y navegar a la carpeta Seccion_3_SQL

```
docker compose up -d
```

- Esperar ~10 segundos a que PostgreSQL inicie

**Paso 2: Ejecutar el script SQL**

- Opcion A - Desde terminal:

```
docker exec -i etl_postgres psql -U etl_user -d etl_prueba
  -f /data/seccion3_sql.sql
```

- Opcion B - Desde DBeaver: Host=localhost, Puerto=5432, BD=etl_prueba
- Usuario=etl_user, Password=etl_pass

**Paso 3: Verificar resultados**

- Carga: FlightsBase=5000, FlightsNew=10000
- Verificacion final: total_registros=10000, registros_unicos=10000
- Texto: "OK - No hay duplicados en FlightsBase"

**Paso 4: Detener el contenedor**

```
docker compose down
```

## Ejecutar Seccion 4 - Logica ETL

- Abrir Seccion_4_Logica_ETL/ETL_Prueba_Tecnica.ipynb en VS Code
- Desplazarse a la seccion "Seccion 4 - Logica ETL"
- El contenido esta en celdas Markdown: Parte A (Diseno), Parte B (Calidad), Parte C (Mejora)

## Resumen de Archivos por Carpeta

**Seccion_1_Excel/**

- flights_5000v2.csv, flights_10000v2.csv, Seccion1_Excel_PruebaTecnica.xlsx

**Seccion_2_Python/**

- ETL_Prueba_Tecnica.ipynb, flights_5000v2.csv, flights_10000v2.csv, flights_unificado_limpio.csv

**Seccion_3_SQL/**

- seccion3_sql.sql, docker-compose.yml, flights_5000.csv, flights_10000.csv

**Seccion_4_Logica_ETL/**

- ETL_Prueba_Tecnica.ipynb