75EC2176_01
Evaluación Continua/Avaluació continua

EC — Normalización y Análisis de Datos de Empleados (PostgreSQL)

0) Contexto y objetivo

Dispones de un CSV con información de empleados (RR. HH.). Tu objetivo es **diseñar** un esquema relacional limpio, **cargar** datos desde CSV, **normalizar** dimensiones de texto, **imponer calidad** con constraints y dominios, y **demostrar consulta analítica** con subconsultas y agregaciones.

Archivo fuente (cabecera esperada)

```
id,name,age,gender,department,job_title,experience_years,edu_level,location,salary
```

Nota: Puede haber variaciones de mayúsculas/minúsculas y espacios extra en los campos textuales.

---

1) Esquema y dominios (10 %)

1. Crea un **schema** exclusivo llamado employees:
2. Define al menos **dos** `DOMAIN` para reforzar calidad:

Entrega: script SQL con creación de schema y dominios.

---

2) Staging y carga (`COPY`) (10 %)

1. Crea una tabla de **staging** `employees.employee_raw` (sin FKs) con columnas equivalentes al CSV.
2. Carga el CSV usando `COPY`
3. Verifica el recuento de filas cargadas.

Entrega: DDL de `employee_raw` + sentencia `COPY` + `SELECT COUNT(*)`.

75EC2176_01
Evaluación Continua/Avaluació continua
3) Normalización de textos (15 %)

1. Estandariza `gender`, `department`, `job_title`, `edu_level`, `location`:
   - Aplica `lower()`, `trim()`, y colapso de espacios (`regexp_replace(..., '\s+', ' ', 'g')`).
2. Crea tablas de **dimensión**:
   - `dim_gender`, `dim_department`, `dim_job_title`, `dim_edu_level`, `dim_location` (id serial + `value` único).
3. **Puebla** cada dimensión con los **DISTINCT** normalizados (usa `INSERT ... ON CONFLICT DO NOTHING`).

Entrega: DDL de dimensiones + DML de inserción normalizada.

---

4) Tabla de hechos y constraints (15 %)

1. Crea `employees.employee` con:
   - `employee_id`
   - `src_id` (id original del CSV) + `name`, `age dom_age`, `experience_years` (`CHECK` entre 0 y 80), `salary dom_money`.
   - Columnas FK: `gender_id`, `department_id`, `job_title_id`, `edu_level_id`, `location_id`.
2. Inserta desde `employee_raw` mapeando a IDs de dimensión mediante **joins** a las tablas dim.
3. Añade **FOREIGN KEY** y **UNIQUE** necesarios (p. ej., `UNIQUE(src_id)` si aplica).

Entrega: DDL de `employee` + DML de inserción con mapeo a dimensiones + `ALTER TABLE ... ADD CONSTRAINT`.

---

5) Índices (10 %)

1. Crea **índices** en:
   - Dimensiones: `value` (búsqueda por etiqueta).
   - Hechos: `department_id`, `job_title_id`, `location_id`, `salary`, `age`.
2. Justifica brevemente (1–2 líneas) por qué cada índice aporta rendimiento para tus consultas del apartado 7.

Entrega: `CREATE INDEX` + breve justificación.

75EC2176_01
Evaluación Continua/Avaluació continua
6) Vistas (10 %)

1. Crea una **vista** `employees.v_employee_dim` que devuelva la "vista amigable" (códigos → etiquetas) con joins a todas las dimensiones.
2. Crea una **vista analítica** adicional (a elegir), por ejemplo:
    - `v_salary_by_department` con `department`, `count`, `avg_salary`, `median` (usa `percentile_cont`).
    - o una vista de **brecha salarial** por `job_title` y `gender`.

Entrega: DDL de las dos vistas.

---

7) Consultas obligatorias (25 %)

Realiza las siguientes consultas **usando la vista** `v_employee_dim` salvo que se indique lo contrario.

7.1 Joins y `ORDER BY`

- Lista de empleados con su `department` y `job_title`, ordenados por `salary` descendente y, a igualdad, por `name`ascendente.

7.2 `GROUP BY ... HAVING`

- Salario medio, mediana (`percentile_cont(0.5)`), y conteo por `department`.
- Filtra con `HAVING` los departamentos con **≥ 10 empleados** y **salario medio ≥ el salario medio global** (subconsulta escalar permitida).

7.3 Subconsulta escalar

- Muestra `name`, `salary` y el **salario medio global** (misma cifra para todas las filas) y un flag `is_above_avg`(`true/false`).

7.4 Subconsulta con `IN`

- Empleados que trabajan en los **top-3** `location` por **número de empleados**. (Obtén primero las 3 `location` con mayor `COUNT(*)` en una subconsulta y utilízala con `IN`.)

75EC2176_01
Evaluación Continua/Avaluació continua

7.5 Subconsulta en `FROM` (tabla derivada)

- Para cada `job_title`, calcula en una subconsulta (tabla derivada) el `avg_salary` y `avg_experience_years`.
  Únela a la vista principal para devolver empleados cuyo `salary` > `avg_salary` de su `job_title`.

7.6 Calidad de datos

- Devuelve **hasta 50** filas con algún **FK nulo** en `employees.employee` (para detectar valores no mapeados o faltantes).
  (Esta consulta se hace sobre la **tabla de hechos**, no sobre la vista.)

Entrega: archivo `.sql` con todas las consultas y resultados (capturas o `\o` en texto).

---

8) Checks y validaciones adicionales (5 %)

- Asegura con `CHECK` que `salary >= 0`, `age` en rango, `experience_years` en rango.
- Añade un `CHECK` opcional para prohibir `salary > 0` cuando `job_title_id` es `NULL` (simula regla de negocio).

Entrega: `ALTER TABLE ... ADD CONSTRAINT ... CHECK`.

---

10) Qué entregar

1. **`01_schema_domains.sql`** — schema + dominios.
2. **`02_staging_copy.sql`** — staging + carga CSV.
3. **`03_dims_fact.sql`** — dimensiones + tabla de hechos + constraints.
4. **`04_indexes_views.sql`** — índices + vistas.
5. **`05_queries.sql`** — consultas del punto 7.
6. **Breve README** (máx. 1 página) con:
   - Suposiciones y decisiones de diseño.
   - Justificación de índices.
   - Cómo ejecutar el proyecto (orden de scripts).
