# JUAN PABLO HOLGUÍN MARÍN

# PRUEBA TÉCNICA

# SECCIÓN: LENGUAJES

Cuáles de los siguientes lenguajes de programación manejas:

* + SQL
  + MySQL
  + R
  + Python
  + DAX
  + Otros:\_PHP\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# SECCIÓN: PRUEBA TÉCNICA SQL

Se tienen 4 tablas de la siguiente manera:

* **Clientes**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Cod | CodSec | Sec | claseEconomica | Cantvisitas | IDDireccion | CodVision |

Donde:

* **ID:** Hace referencia al identificador de cada cliente dentro de la empresa.
* **Cod:** Indica el Código interno al que pertenece el cliente
* **CodSec:** Hace referencia al código del tipo de cliente.
* **Sec:** descripción del tipo de cliente.
* **claseEconomica:** descripción actividad del cliente.
* **Cantvisitas**: Visitas hechas al cliente.
* **IDDireccion**: Identificador dirección.
* **CodVision**: Código interno ventas.
* **Territorio\_1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IDDireccion2 | MunicipioDireccion | LongitudDireccion | LatitudDireccion | IDTerritorio |

Donde:

* **IDDireccion2:** Identificador direccion.
* **MunicipioDireccion:** Indica el municipio de la dirección.
* **LongitudDireccion:** Longitud.
* **LatitudDireccion:** Latitud.
* **IDTerritorio:** identificador territorio.
* **Territorio\_2**

|  |  |  |  |
| --- | --- | --- | --- |
| IDTerritorio | ComunaTerritorio | Territorio | SubregionTerritorio |

Donde:

* **IDTerritorio:** Identificador territorio.
* **ComunaTerritorio:** Indica la comuna del territorio.
* **Territorio:** territorio de la comuna.
* **SubregionTerritorio:** subregión de la comuna.
* **Venta**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| fecha\_fact | fact | fact\_pos | canal | ID | denominacion\_1 | valor | servicio |
| descservicio | tiposervicio | tipopres |

* **fecha\_fact:** hace referencia a fecha factura. Formato DD/MM/AAAA
* **fact:** Número factura.
* **fact\_pos:** Posición factura.
* **canal:** Código interno venta.
* **ID:** Indica el ID del cliente.
* **denominación\_1**: Indica tipo factura
* **valor**: Valor de la factura
* **servicio**: Servicio prestado general.
* **descservicio**: Descripción del servicio prestado.
* **Tiposervicio**: Tipo de venta interna o externa.
* **Tipopres**: Servicio especial o normal.

## Escriba en lenguaje SQL la respuesta a las siguientes preguntas:

1. ¿Cuántos clientes únicos se tienen?

SELECT COUNT(DISTINCT "ID") AS TotalClientes

FROM Clientes;

1. Muestre la cantidad de clientes que hay por claseEconomica.

SELECT "claseEconomica", COUNT(\*) AS CantidadClientes

FROM Clientes

GROUP BY "claseEconomica"

ORDER BY CantidadClientes desc;

1. Muestre cuantos tipos de clientes se tienen (Sec).

SELECT COUNT(DISTINCT "Sec") AS TiposClientes

FROM Clientes;

SELECT "Sec", COUNT(\*) AS NumeroClientes

FROM Clientes

GROUP BY "Sec"

ORDER BY NumeroClientes DESC;

1. ¿Cuál es el tipo de cliente (Sec) que más visita acumula?

SELECT "Sec", SUM("Cantvisitas") AS TotalVisitas

FROM Clientes

GROUP BY "Sec"

ORDER BY TotalVisitas DESC

--LIMIT 1;

1. Obtenga la tabla con las siguientes columnas: ID, tipo de cliente, visitas, MunicipioDireccion. **Si un cliente no tiene el Municipio se debe traer Nulo el valor.**

SELECT DISTINCT "c"."ID", "c"."Sec" AS "tipo de cliente", "c"."Cantvisitas" AS "visitas", COALESCE("t1"."MunicipioDireccion", 'Nulo') AS "MunicipioDireccion"

FROM Clientes AS "c"

LEFT JOIN Territorio\_1 AS "t1" ON "c"."IDDireccion" = "t1"."IDDireccion2";

1. Obtenga la tabla con las siguientes columnas ID, tipo de cliente, visitas, MunicipioDireccion, SubregionTerritorio. **En esta tabla no se deben traer Nulos en las columnas de MunicipioDireccion, SubregionTerritorio**

SELECT DISTINCT ON ("c"."ID")

"c"."ID",

"c"."Sec" AS "tipo de cliente",

"c"."Cantvisitas" AS "visitas",

"t1"."MunicipioDireccion",

"t2"."SubregionTerritorio"

FROM Clientes AS "c"

INNER JOIN Territorio\_1 AS "t1" ON "c"."IDDireccion" = "t1"."IDDireccion2"

INNER JOIN Territorio\_2 AS "t2" ON "t1"."IDTerritorio" = "t2"."IDTerritorio"

WHERE "t1"."MunicipioDireccion" IS NOT NULL

AND "t2"."SubregionTerritorio" IS NOT NULL

ORDER BY "c"."ID", "t1"."MunicipioDireccion", "t2"."SubregionTerritorio";

1. Muestre el top 10 de clientes (ID) con el mayor valor de compra.

SELECT "ID", SUM("valor") AS "TotalCompra"

FROM Venta

GROUP BY "ID"

ORDER BY "TotalCompra" DESC

LIMIT 10;

# SECCIÓN: PRUEBA TÉCNICA POWER BI

EL grupo primario de lideres te solicita realizar un tablero y análisis del comportamiento observado para la toma de decisiones, para esto es necesario utilizar PowerBI. El documento con el análisis debe ser de una página.

Recuerda que cuentas con 4 archivos para realizar este ejercicio de entendimiento del cliente, donde se tiene su descripción general, su territorio (2) y las ventas a los mismos.

Recomendaciones:

1. Realice la limpieza necesaria para poder usar la información suministrada (bonus: Python)
2. Adjunte pantallazos del proceso de generación del tablero.
3. Comparta el archivo .pbix
4. Utilice los gráficos y clores que considere necesarios para este ejercicio.
5. Recuerde el público objetivo del tablero para desarrollar el análisis.

**SOLUCIÓN**

Para llevar a cabo este ejercicio se partió de la construcción de una bodega de datos que almacena los datos procesados provenientes de un lago de datos en donde reposan los archivos CSV originales. Por medio de un pipeline se ejecutan funciones de Extracción, Transformación y Carga de los archivos csv, lo cual permite realizar el procesamiento de los mismos y finalmente cargarlos en un gestor de bases de datos, en este caso, PostgreSQL.

Crear una bodega de datos mediante un proceso ETL con Python para extraer archivos CSV, transformarlos, cargarlos en PostgreSQL y luego construir un tablero de Power BI conectado a PostgreSQL ofrece grandes beneficios en términos de calidad de datos, automatización y eficiencia. Esta metodología permite limpiar y validar datos, automatizar la actualización de información, y centralizar datos de múltiples fuentes, mejorando la integridad y disponibilidad de la información. Además, facilita el análisis avanzado y la generación de reportes interactivos en Power BI, apoyando la toma de decisiones basada en datos confiables y actualizados.

La solución a este ejercicio se presenta en un repositorio de Github que se anexa, en donde se adjunta el código de Python para la construcción de la bodega de datos, el tablero de power Bi, las consultas en SQL y este archivo de Word con la descripción del proceso.