Base de Datos I
Trabajo Práctico Especial
1er Cuatrimestre 2019

1. *Objetivo*

El objetivo de este Trabajo Práctico Especial es aplicar los conceptos de SQL Avanzado (PSM, Triggers) vistos a lo largo del curso, para implementar funcionalidades y restricciones no disponibles de forma estándar (que no pueden resolverse con Primary Keys, Foreign Keys, etc.).

2. Modalidad

El Trabajo Práctico estará disponible en el Campus a partir del 06/06/2019, indicándose allí mismo la fecha de entrega.

Se incluye junto con el enunciado el archivo: **SalesbyRegion.csv**.

El TP deberá realizarse en grupos de 3 alumnos y entregarse a través de la plataforma Campus ITBA hasta la fecha allí indicada.

3. Descripción del Trabajo

Para este trabajo hemos decidido usar un archivo CSV (Comma Separated Values).

El archivo **SalesbyRegion.csv** contiene información de Ventas, tales como Ingresos (Revenue) y Costos (Cost), de una empresa de Retail agrupada por diferentes dimensiones:

- Trimestre (Quarter)
- Mes (Month)
- Semana (Week)
- Tipo de producto (Product Type)
- Región (Territory)
- Canal de venta (Sales Channel)
- Tipo de cliente (Customer Type)

La finalidad de este Trabajo Práctico Especial consiste en migrar los datos del archivo csv a una base de datos, generar un cálculo, producir un reporte y realizar algunas validaciones. Específicamente se debe hacer lo siguiente:

a) Crear una tabla intermedia para importar los datos
b) Crear la tabla definitiva donde los campos Quarter, Month y Week, se conviertan en un único campo fecha del tipo Date
c) Importar los datos
d) Calcular el promedio del margen de venta para N meses móvil para atrás
e) Crear un reporte de ventas histórico

a) Creación de la tabla intermedia.

Debe crearse un tabla intermedia llamada **INTERMEDIA** que será la receptora de datos del archivo **SalesbyRegion.csv**. Los campos y restricciones de la tabla

deben crearse en base al análisis de los datos. Recordar que los archivos csv son archivos de texto que pueden abrirse fácilmente con cualquier editor. Se recomienda que los nombres de los campos coincidan con los de las columnas del archivo csv, cambiando los espacios por _.

La sentencia CREATE TABLE quedaría de la siguiente manera:

```
CREATE TABLE intermedia(
      Quarter TEXT NOT NULL,
      Month TEXT NOT NULL,
      Week TEXT NOT NULL,
      Product_Type TEXT NOT NULL,
      Territory TEXT NOT NULL,
      Sales_Channel TEXT NOT NULL,
      Customer_Type TEXT NOT NULL,
      Revenue FLOAT,
      Cost FLOAT
);
```

En base a los datos, se deben crear las claves y constraints apropiados.

b) **Creación de la tabla definitiva.**

Debe crearse un tabla definitiva llamada **DEFINITIVA** que será completada a medida que se inserten los datos en la tabla INTERMEDIA. Los campos de la tabla son iguales a la tabla INTERMEDIA, a diferencia de que los campos Quarter, Month y Week se reemplazan por el campo Sales_Date, el cual será un campo del tipo DATE YYYY-MM-DD (donde DD = 01).

En resumen, los campos de la tabla DEFINITIVA deben ser como mínimo los siguientes:

- Sales_Date
- Product_Type
- Territory
- Sales_Channel
- Customer_Type
- Revenue
- Cost

Tal como se verá en el punto c), se utilizarán Triggers para cargar los datos de esta tabla.

c) **Importación de los datos**

Utilizando el comando COPY de PostgreSQL, se deben importar TODOS los datos del archivo **csv** en la tabla creada en **a)**. Los datos del archivo **csv** provisto por la cátedra NO puede ser modificado.

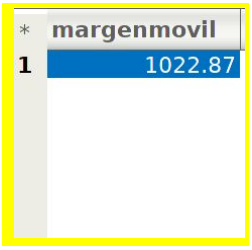
A medida que los datos son insertados en la tabla INTERMEDIA, deben insertarse automáticamente en la tabla DEFINITIVA. Para ello hay crear el Trigger **InsertaDefinitiva** que ante una inserción en la tabla INTERMEDIA inserte una tupla en la tabla DEFINITIVA con los cambios apropiados (explicados en b).

d) **Cálculo del margen de venta promedio móvil**

En este caso queremos calcular el promedio móvil del margen de venta (Revenue – Cost). Es decir, se pide crear una función **MargenMovil(fecha,n)** que recibe como parámetros una fecha dada del tipo DATE y una cantidad entera de meses, para calcular el promedio del margen de venta total de los últimos n meses, tomando como fecha pivot a la fecha ingresada como parámetro.

Por ejemplo,

- si invocamos **MargenMovil(to_date('2012-11-01','YYYY-MM-DD'),3)** se debe obtener:



*	margenmovil
1	1022.87

- Si invocamos **MargenMovil(0)**, no se debe mostrar nada y enviar un mensaje a la consola que diga "La cantidad de meses anteriores debe ser mayor a 0"

e) **Reporte de Ventas Histórico.**

Una vez al mes, el gerente de la empresa de Retail controla las ventas históricas agrupadas por Año, por los distintos Canales de Ventas y por los distintos Tipos de Cliente.

Se pide crear la función **ReporteVentas(n)** que recibe como parámetro la cantidad de años enteros a mostrar sobre la tabla **DEFINITIVA**, la cual genere un reporte mostrando para cada Año, los Ingresos, los Costos y el Margen.

El reporte tendrá las siguientes características:

I. Encabezado con título "**HISTORIC SALES REPORT**"

II. Encabezado de columnas:

```
"Year      Category     Revenue      Cost  Margin"
```

III. Por cada año tiene que aparecer un renglón en el reporte, con el Año **ordenados de menor a mayor**. La primer categoría de agrupación (Sales Channel) ordenadas alfabéticamente y sus métricas (Revenue, Cost y Margin: Revenue-Cost), deben estar en el **mismo** renglón que el año y el resto de las categorías (Sales Channel y Customer Type), encolumnados a continuación en los renglones subsiguientes

IV. Al final de todo, tiene que aparecer el total del año

En caso de que no existieran datos para los parámetros ingresados, no se debe mostrar nada (ni siquiera el encabezado).

La función debe manejar los posibles errores.

Para saber cómo escribir el reporte ver el apéndice Cómo escribir por pantalla.

Por ejemplo,

- si invocamos **ReporteVentas(1)** se debe obtener:

HISTORIC SALES REPORT

YEAR	CATEGORY	REVENUE	COST	MARGIN
2011	Sales Channel: Direct	6066302	5579294	487008
	Sales Channel: Internet	6474938	4606461	1854770
	Sales Channel: Retail	6144569	7184209	-1039640
	Customer Type: Branded Stores	2844339	2767014	66829
	Customer Type: Entertainment Venues	6129525	4287749	1852317
	Customer Type: Internet Direct	2953491	1454881	1498610
	Customer Type: Retail	6648327	8871998	-2124085
	Total:	18685809	17381642	1304167

- Si invocamos **ReporteVentas(3)** se debe obtener obtener lo mismo que invocando **ReporteVentas(2)**

- Si invocamos **ReporteVentas(0)** no se obtiene nada

4. Entregables

Los alumnos deberán entregar los siguientes documentos:

- El script sql **funciones.sql** con el código necesario para crear las tablas, las funciones y los triggers

- Un informe que debe contener:
  - El rol de cada uno de los participantes del grupo. Si bien en el TP deben estar involucrados todos los integrantes, se debe asignar un rol de supervisión de cada una de las tareas. Mínimamente los roles son: encargado del informe, encargado de las funciones, encargado del trigger, encargado del funcionamiento global del proyecto y encargado de investigación. Pueden asignarse más roles en caso de requerirse
  - Todo lo investigado para realizar el TP
  - Las dificultades encontradas y cómo se resolvieron
  - También se debe detallar aquí el proceso de importación de los datos realizado
  - El informe debe tener como máximo 3 páginas

5. Evaluación

La evaluación del trabajo se llevará a cabo utilizando los parámetros establecidos en la rúbrica asociada a la actividad en el Campus.

Se tendrá en cuenta que las consultas, más allá del funcionamiento (lo cual es fundamental), sean genéricas.

Los docentes ejecutarán el proceso usando los conjuntos de datos entregados pero podrán también hacer pruebas con otros conjuntos de datos de similares características para evaluar el funcionamiento en distintos escenarios.

El informe deberá estar completo y sin faltas de ortografía.

En caso de que el trabajo no cumpliera los requisitos básicos para ser aprobado, los alumnos serán citados en la fecha de recuperatorio para defenderlo y corregir los errores detectados.

Apéndices

Cómo escribir en pantalla

En el servidor bd1.it.itba.edu.ar tenemos instalada la extensión **dbms_output**. Este paquete de Oracle agrega funcionalidad a PostgreSQL permitiéndole escribir por pantalla. Las funciones disponibles son:

Funciones

dbms_output.enable([buffer_size int4])	Start dbms_output support, elective buffer_size set up maximum buffer storage size
dbms_output.disable()	Deactivate dbms_output support, put, put_line, new_line commands have no effect
dbms_output.serveroutput(bool)	Start client output display demand, start dbms_output at the same time
dbms_output.put(text)	Insert text in output buffer
dbms_output.put_line(text)	Insert line (text with end of line symbol)
dbms_output.new_line()	Insert end of line symbol

Por ejemplo, para imprimir por pantalla ‘Hello World!’, se debe ejecutar:

```
DO $$
BEGIN
        PERFORM DBMS_OUTPUT.DISABLE();
        PERFORM DBMS_OUTPUT.ENABLE();
        PERFORM DBMS_OUTPUT.SERVEROUTPUT ('t');
        PERFORM DBMS_OUTPUT.PUT_LINE ('Hello World!');
END; $$
```

Se recomienda hacer este trabajo en el servidor y no localmente dado que esta extensión no se encuentra disponible por default.