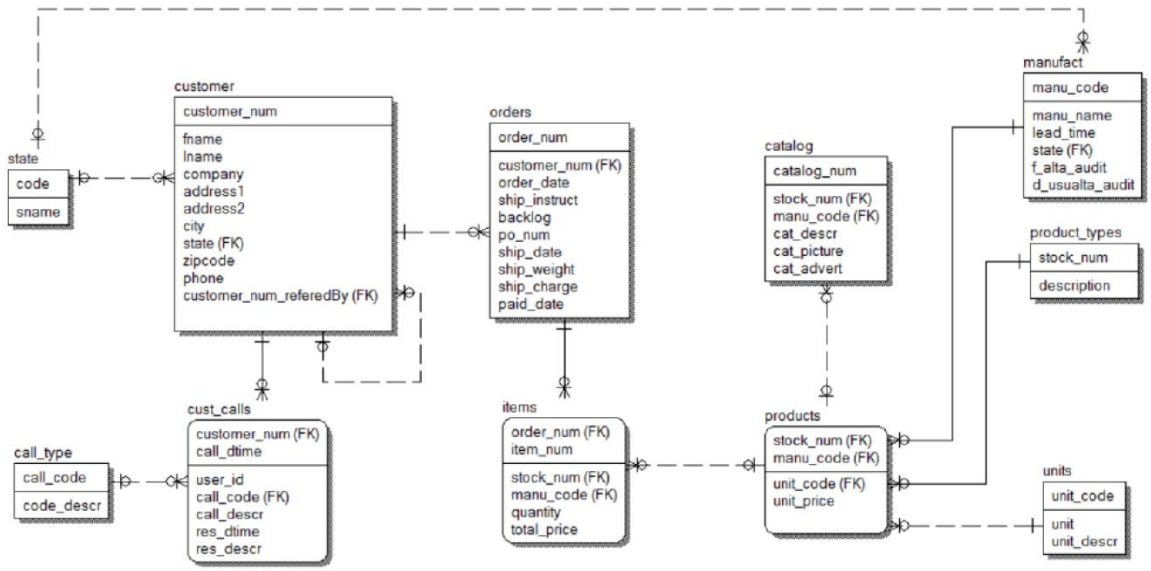
Práctica de SQL

Modelo de Datos BD stores7



JOINS (Group by, having, Subqueries, subq. correlacionados, outer joins, Temp tables)

1. Mostrar una lista de todos los fabricantes con el monto total de las ventas realizadas, ordenado por código de fabricante. En el caso de no tener ventas se deberá poner el total de ventas en NULO.
   El listado debe contener:
   - Código de fabricante (manu_code)
   - Nombre de fabricante (manu_name)
   - Lead Time (lead_time)
   - Monto (Sumatoria del cambo total_price de la tabla items)

```
SELECT i.manu_code, manu_name, lead_time, SUM(total_price)
FROM items i RIGHT JOIN manufact m ON (i.manu_code=m.manu_code)
GROUP BY i.manu_code,manu_name, lead_time
```

2. Mostrar una lista de a pares, de todos los fabricantes que fabriquen el mismo producto. En el caso que haya un único fabricante deberá mostrar el Código de fabricante 2 en nulo.

   El listado tiene que tener el siguiente formato:

Nro. de Producto (stock_num)	Descripc. del producto (Description)	Cód. de fabric. 1 (manu_code)	Cód. de fabric. 2 (manu_code)

```
SELECT s1.stock_num,tp.description,s1.manu_code, s2.manu_code
FROM products s1 LEFT JOIN products s2
ON (s1.stock_num=s2.stock_num AND s1.manu_code != s2.manu_code)
      JOIN product_Types tp ON (s1.stock_num = tp.stock_num)
```

3. Listar todos los clientes que hayan puesto más de una orden.
   a) En primer lugar, escribir una consulta usando una subconsulta.
   b) Reescribir la consulta usando dos sentencias SELECT y una tabla temporal.
   c) Reescribir la consulta utilizando GROUP BY y HAVING.

   La consulta deberá tener el siguiente formato:

Número_de_Cliente (customer_num)	Nombre (fname)	Apellido (lname)

a.1

```
SELECT customer_num, lname,fname
FROM customer
WHERE customer_num IN (SELECT customer_num FROM orders
                  GROUP BY customer_num HAVING COUNT(order_num)>1)
```

a2

```
SELECT customer_num, lname,fname
FROM customer c
WHERE EXISTS (SELECT customer_num FROM orders o
                      WHERE o.customer_num = c.customer_num
                  GROUP BY customer_num HAVING COUNT(order_num)>1)
```

a3

```
SELECT customer_num, lname,fname
FROM customer c
WHERE (SELECT COUNT(order_num) FROM orders o
        WHERE o.customer_num = c.customer_num)>1
```

a4

```
SELECT c.customer_num, lname,fname
FROM customer c JOIN (SELECT customer_num FROM orders
             GROUP BY customer_num HAVING COUNT(order_num)>1) sub1
             ON c.customer_num = sub1.customer_num
```

b

```
SELECT customer_num
INTO #clieTemp
FROM orders
GROUP BY customer_num HAVING COUNT(order_num)>1

SELECT c.customer_num, lname,fname
FROM customer c JOIN #clieTemp c2
             ON c.customer_num = c2.customer_num
```

C

```
SELECT c.customer_num, lname,fname
FROM customer c JOIN orders o
ON (c.customer_num=o.customer_num)
GROUP BY c.customer_num, lname, fname
HAVING COUNT(order_num)>1
```

4. Encontrar todas las Órdenes de compra con el Monto total (Suma del total_price de sus items) menor que el precio total promedio (total_price) de todos los ítems de todas las ordenes.

Formato de la salida:	Nro. de Orden (order_num)	Total (suma)

```
SELECT o.order_num 'Nro. de Orden', SUM(total_price) Total
FROM orders o JOIN items I ON (o.order_num = i.order_num)
GROUP BY o.order_num
HAVING SUM(total_price)<(SELECT AVG(total_price) FROM items)
```

5. Se quiere obtener por cada fabricante, el listado de todos los productos de stock con precio unitario (unit_price) mayor que el precio unitario promedio para dicho fabricante. El campos de salida serán: manu_code, manu_name, stock_num, description, unit_price.

   Por ejemplo:

   El precio unitario promedio de los items fabricados por ANZ es $180.23. se debe incluir en su lista todos los items de stock de ANZ que tengan un precio unitario superior que dicho importe.

```
SELECT s.manu_code, manu_name, s.stock_num, description, unit_price
FROM products s JOIN manufact m ON (s.manu_code=m.manu_code)
JOIN product_Types tp ON (tp.stock_num = s.stock_num)
WHERE unit_price > (SELECT AVG(unit_price) FROM products s2
WHERE s2.manu_code = m.manu_code)
```

6. Usando el operador NOT EXISTS listar la información de órdenes de compra que no incluyan ningún producto que contenga en su descripción el string ‘ baseball gloves’. Ordenar los resultados por compañía del cliente ascendente y número de orden descendente.

   El formato de salida deberá ser:

Número de Cliente (customer_num)	Compañía (company)	Número de Orden (order_num)	Fecha de la Orden (order_date)

```
SELECT c.customer_num, company, o.order_num, order_date
FROM orders o JOIN customer c ON (o.customer_num=c.customer_num)
WHERE NOT EXISTS (SELECT item_num FROM items i JOIN product_Types tp
                    ON (i.stock_num=tp.stock_num)
                  WHERE description LIKE '%baseball gloves%'
                    AND i.order_num = o.order_num)
ORDER BY order_num asc
```

Operador UNION

7. Reescribir esta consulta usando el operador UNION:
   SELECT * FROM stock
   WHERE manu_code = “HRO”
   OR stock_num = 1

```
SELECT * FROM stock WHERE manu_code='HRO'
UNION
SELECT * FROM stock WHERE stock_num=1
```

8. Desarrollar una consulta que devuelva la siguiente información de la tabla clientes (customer) ordenadas por ciudad (city). Asimismo, ubicar a los clientes de Redwood City al principio de la lista y el resto de los clientes pertenecientes a otras ciudades ordenarlos alfabéticamente por ciudad.

   Formato: Clave de ordenamiento (sortkey) | Ciudad (city) | Compañía (company)

```
SELECT 1 sortkey,city ciudad, Company compañía
FROM customer
WHERE city ='Redwood City'
UNION
SELECT 2 sortkey,city ciudad, Company compañía
FROM customer
WHERE city !='Redwood City'
ORDER BY sortkey,city
```

Vistas

9. Crear una Vista llamada ClientesConMultiplesOrdenes basada en la consulta realizada en el punto 3.c con los nombres de atributos solicitados en dicho punto.

```
CREATE VIEW ClientesConMultiplesOrdenes
(numero_cliente, nombre, apellido)
AS
SELECT c.customer_num, lname,fname
FROM customer c JOIN orders o
ON (c.customer_num=o.customer_num)
GROUP BY c.customer_num, lname, fname
HAVING COUNT(order_num)>1
```

10. Crear una Vista llamada Productos_HRO en base a la consulta

    SELECT * FROM stock
    WHERE manu_code = “HRO”

    , la vista deberá restringir la posibilidad de insertar datos que no cumplan con su criterio de selección.

    a. Realizar un INSERT de un Producto con manu_code=’ANZ’. Qué sucede?
    b. Realizar un INSERT con manu_code=’HRO’ y stock_num=100. Qué sucede?
    c. Validar los datos insertados a través de la vista.

```
CREATE VIEW Productos_HRO
AS
SELECT * FROM products
```

```
WHERE manu_code = 'HRO'
WITH CHECK OPTION

INSERT INTO Productos_HRO (stock_num,manu_code)
VALUES (303,'ANZ')

INSERT INTO Productos_HRO (stock_num,manu_code)
 VALUES (303,'HRO')
```

Transacciones

11. Escriba una transacción que incluya las siguientes acciones:
    - BEGIN TRANSACTION
        - Insertar un nuevo cliente llamado “Fred Flintstone” en la tabla de clientes (customer).
        - Seleccionar todos los clientes llamados Fred de la tabla de clientes (customer).
    - ROLLBACK TRANSACTION

    Luego volver a ejecutar la consulta
    - Seleccionar todos los clientes llamados Fred de la tabla de clientes (customer).
    - Completado el ejercicio descripto arriba. Observar que los resultados del segundo SELECT difieren con respecto al primero.

BEGIN TRANSACTION
INSERT INTO customer VALUES ()
SELECT ….

ROLLBACK TRANSACTION

SELECT …

12. Se ha decidido crear un nuevo fabricante AZZ, quién proveerá parte de los mismos productos que provee el fabricante ANZ, los productos serán los que contengan el string ‘tennis’ en su descripción.
    - El código del nuevo fabricante será “AZZ”, el nombre de la compañía “AZZIO Division SA” y el tiempo de envío será de 5 días (lead_time).
    - Usted tendrá que agregar la nuevas filas en la tabla manufact y la tabla products.
    - La información de la tabla products será la misma que la actual del fabricante “ANZ”, para el nuevo fabricante “AZZ” pero sólo de los productos que contengan 'tennis' en su descripción.
    - Tener en cuenta las restricciones de integridad referencial existentes, manejar todo dentro de una misma transacción.

BEGIN TRANSACTION
INSERT INTO customer VALUES ()

SELECT ….

ROLLBACK TRANSACTION

SELECT …