* 1. **Stored Procedures**

1. **Crear la siguiente tabla CustomerStatistics con los siguientes campos customer\_num (entero y pk), ordersqty (entero), maxdate (date), uniqueProducts (entero)**
2. **Crear un procedimiento ‘CustomerStatisticsUpdate’ que reciba el parámetro fecha\_DES (fecha\_DES) y que en base a los datos de la tabla customer, inserte (si no existe) o actualice el registro de la tabla CustomerStatistics con la siguiente información:**

**Ordersqty: cantidad de órdenes para cada cliente + las nuevas órdenes con fecha mayor o igual a fecha\_DES**

**Maxdate: fecha máxima de la última órden puesta por cada cliente.**

**uniqueProducts: cantidad única de productos adquiridos por cada cliente histórica**

drop table CustomerStatistics;

create table CustomerStatistics

(customer\_num integer primary key,

ordersqty integer,

maxdate datetime,

uniqueManufact integer);

SELECT \* FROM sys.columns WHERE object\_id = OBJECT\_ID('CustomerStatistics');

drop procedure CustomerStatisticsUpdate;

CREATE PROCEDURE CustomerStatisticsUpdate @fecha\_DES DATETIME AS

BEGIN

DECLARE CustomerCursor

CURSOR FOR SELECT customer\_num from customer;

DECLARE @customer\_num INT, @ordersqty INT, @maxdate DATETIME, @uniqueManufact INT;

OPEN CustomerCursor;

FETCH NEXT FROM CustomerCursor INTO @customer\_num

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @ordersqty=count(\*) , @maxDate=max(order\_date)

FROM orders

WHERE customer\_num = @customer\_num

AND order\_date >= @fecha\_DES;

SELECT @uniqueManufact=count(distinct stock\_num)

FROM items i, orders o

WHERE o.customer\_num = @customer\_num

AND o.order\_num = i.order\_num;

IF NOT EXISTS( SELECT 1 FROM CustomerStatistics

WHERE customer\_num = @customer\_num)

INSERT into customerStatistics VALUES (@customer\_num, @ordersQty, @maxDate, @uniqueManufact);

ELSE

UPDATE customerStatistics

SET ordersQty=ordersQty+@ordersQty,

maxDate=@maxDate,

uniqueManufact= @uniqueManufact

WHERE customer\_num = @customer\_num;

FETCH NEXT FROM CustomerCursor INTO @customer\_num

END;

CLOSE CustomerCursor;

DEALLOCATE CustomerCursor;

END;

--Para probarlo

insert into customer (customer\_num,lname,fname)

values (10000,'zzz','zzz')

insert into orders (order\_num, order\_date, customer\_num)

values (10001,'2017-10-24',10000)

declare @fecha datetime

set @fecha = '2017-01-01'

exec CustomerStatisticsUpdate @fecha

select @fecha

select \* from CustomerStatistics

--------------------------

**--Otra alternativa**

CREATE OR ALTER PROCEDURE dbo.CustomerStatisticsUpdate (@fecha\_DES DATE) AS

BEGIN

MERGE CustomerStatistics tD

USING (SELECT c.customer\_num, COUNT(DISTINCT o.order\_num) ordersQty,

MAX(o.order\_date) maxdate, COUNT(DISTINCT i.stock\_num) uniqueManufact

FROM customer c LEFT JOIN orders o on c.customer\_num = o.customer\_num

LEFT JOIN items i on o.order\_num = i.order\_num

WHERE o.order\_date > @fecha\_DES -- OR o.order\_date IS NULL

GROUP BY c.customer\_num) tF

ON (tD.customer\_num = tF.customer\_num)

WHEN MATCHED then

update set td.ordersQty = td.ordersQty + tf.ordersQty,

td.maxDate = tf.maxDate,

td.uniqueManufact = tf.uniqueManufact

WHEN NOT MATCHED BY TARGET THEN

INSERT (customer\_num, ordersqty, maxdate, uniqueManufact)

VALUES (tf.customer\_num, tf.ordersqty, tf.maxdate, tf.uniqueManufact);

END

declare @fecha datetime

set @fecha = '2017-01-01'

exec CustomerStatisticsUpdate @fecha

select \* from CustomerStatistics

* 1. **Stored Procedures**

1. **Crear la siguiente tabla informeStock con los siguientes campos fechaInforme (date), stock\_num (entero), manu\_code (char(3) ), cantOrdenes (entero), UltCompra (date), cantClientes (entero), totalVentas (decimal). PK (fechaInforme, stock\_num, manu\_code)**
2. **Crear un procedimiento ‘generarInformeGerencial’ que reciba un parámetro fechaInforme y que en base a los datos de la tabla PRODUCTS de todos los productos existentes, inserte un registro de la tabla informeStock con la siguiente información:**

**fechaInforme: fecha pasada por parámetro**

**stock\_num: número de stock del producto**

**manu\_code: código del fabricante**

**cantOrdenes: cantidad de órdenes que contengan el producto.**

**UltCompra: fecha de última orden para el producto evaluado.**

**cantClientes: cantidad de clientes únicos que hayan comprado el producto.**

**totalVentas: Sumatoria de las ventas de ese producto (total\_price)**

**Validar que no exista en la tabla informeStock un informe con la misma fechaInforme recibida por parámetro.**

create table informeStock

(fechaInforme datetime,

stock\_num integer,

manu\_code char(3),

cantOrdenes integer,

UltCompra datetime,

cantClientes integer,

totalVentas integer,

PRIMARY KEY (fechaInforme, stock\_num, manu\_code))

CREATE or ALTER PROCEDURE generarInformeGerencial(@fechaInforme date)

AS

BEGIN

IF NOT EXISTS(SELECT \* FROM informeStock

WHERE fechaInforme = @fechaInforme)

BEGIN

INSERT INTO informeStock

SELECT @fechaInforme, s.stock\_num, m.manu\_code, COUNT(o.order\_num), MAX(o.order\_date), COUNT(DISTINCT c.customer\_num), SUM(i.total\_price)

FROM products s JOIN manufact m ON (s.manu\_code = m.manu\_code)

JOIN items i ON (i.stock\_num = s.stock\_num

AND s.manu\_code = i.manu\_code)

JOIN orders o ON (o.order\_num = i.order\_num)

JOIN customer c ON (c.customer\_num = O.customer\_num)

GROUP BY s.stock\_num, m.manu\_code

END

else

RAISERROR('ERROR¡!!FECHA YA GENERADA', 16, 1) -- WITH LOG

END;

declare @fecha datetime

set @fecha = '2017-01-01'

EXEC generarInformeGerencial @fecha

print 'sigue'

-- select \* from informeStock

* 1. **Crear un procedimiento ‘generarInformeVentas’ que reciba un parámetro fechaInforme y codEstado y que en base a los datos de la tabla customer de todos los clientes que vivan en el estado pasado por parámetro, inserte un registro de la tabla informeVentas con la siguiente información:**

**fechaInforme: fecha pasada por parámetro**

**codEstado: código de estado recibido por parámetro**

**customer\_num: número de cliente**

**cantOrdenes: cantidad de órdenes del cliente.**

**primerVenta: fecha de la primer orden al cliente.**

**UltVenta: fecha de última orden al cliente.**

**cantProductos: cantidad de productos únicos que hayan comprado el cliente.**

**totalVentas: Sumatoria de las ventas de ese producto (total\_price)**

**Validar que no exista en la tabla informeVentas un informe con la misma fechaInforme y estado recibido por parámetro.**

drop table informeVENTAS

create table informeVENTAS

(fechaInforme datetime,

cod\_Estado integer,

customer\_num smallint,

cantOrdenes integer,

UltCompra datetime,

cantClientes integer,

totalVentas integer,

PRIMARY KEY (fechaInforme, cod\_Estado,customer\_num))

DROP PROCEDURE generarInformeVentas

CREATE PROCEDURE generarInformeVentas @fechaInforme date, @cod\_Estado smallint AS

BEGIN

IF NOT EXISTS(SELECT \* FROM informeVENTAS

WHERE fechaInforme = @fechaInforme

AND cod\_Estado = @cod\_Estado)

BEGIN

INSERT INTO informeVENTAS

SELECT @fechaInforme,@cod\_Estado, o.customer\_num,

COUNT(o.order\_num),

MAX(o.order\_date), COUNT(DISTINCT p.stock\_num),

SUM(i.total\_price)

FROM products p JOIN manufact m

ON (p.manu\_code = m.manu\_code)

JOIN items AS i ON (i.stock\_num = p.stock\_num

AND i.manu\_code = m.manu\_code)

JOIN orders AS o ON (o.order\_num = i.order\_num)

JOIN customer AS c ON (c.customer\_num = o.customer\_num)

GROUP BY o.customer\_num;

END

END;

declare @fecha datetime

set @fecha = getDate()

EXEC generarInformeVentas @fecha,10

select \* from informeVENTAS