a. Stored Procedures

1. **Crear la tabla CustomerStatistics con los siguientes campos customer_num (entero y pk), ordersQty (entero), maxDate (date), productsQty (entero)**

2. **Crear un procedimiento 'CustomerStatisticsUpdate' que reciba el parámetro fecha_DES (date) 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 de la última órden del cliente.
   productsQty: cantidad única de productos adquiridos por cada cliente histórica

```
drop table CustomerStatistics;

create table CustomerStatistics
(customer_num integer primary key,
 ordersqty integer,
 maxdate datetime,
 productsQty integer);

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, @productsQty
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 @productsQty = count(*)
               FROM (select distinct stock_num, manu_code
                       from items i join orders o
                                    on o.order_num = i.order_num
                   WHERE o.customer_num = @customer_num) A;

        IF NOT EXISTS( SELECT 1 FROM CustomerStatistics
                        WHERE customer_num = @customer_num)
            INSERT into customerStatistics (customer_num, ordersqty, maxdate,
                                            productsQty)
             VALUES (@customer_num, @ordersQty, @maxDate, @productsQty);
        ELSE
             UPDATE customerStatistics
                SET ordersQty=ordersQty+@ordersQty,
                    maxDate=@maxDate,
                    productsQty= @productsQty
              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
```

b. Stored Procedures

1. **Crear la 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 (p x q)

   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))

-- drop PROCEDURE generarInformeGerencial

CREATE PROCEDURE generarInformeGerencial(@fechaInforme datetime)
AS
BEGIN
    IF EXISTS(SELECT * FROM informeStock WHERE fechaInforme = @fechaInforme)
        THROW 50000, 'Mes ya procesado', 1

    INSERT INTO informeStock (fechaInforme, stock_num, manu_code,
                    cantOrdenes, UltCompra, cantClientes, totalVentas)
```

```
        SELECT @fechaInforme, s.stock_num, s.manu_code,
               COUNT(distinct o.order_num), MAX(o.order_date),
               COUNT(DISTINCT o.customer_num),
               SUM(i.unit_price*i.quantity)
         FROM products s LEFT 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)
        GROUP BY s.stock_num, s.manu_code

END;

declare @fecha datetime
set @fecha = '2017-01-01'
EXEC generarInformeGerencial @fecha
-- select * from informeStock
```

c. Crear un procedimiento 'generarInformeVentas' que reciba como parámetros 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.
primerCompra: fecha de la primer orden al cliente.
CantProductos: cantidad de tipos de productos únicos que haya comprado el cliente.
totalCompras: Sumatoria de compras del cliente (p x q)

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     char(2),
 customer_num   smallint,
 cantOrdenes    integer,
 primerCompra   datetime,
 cantProductos  integer,
 totalCompras    integer,
     PRIMARY KEY (fechaInforme, cod_Estado, customer_num));

CREATE PROCEDURE generarInformeVentas @fechaInforme date, @cod_Estado char(2) AS
BEGIN
       IF EXISTS(SELECT * FROM informeVENTAS
                  WHERE fechaInforme = @fechaInforme
                       AND cod_Estado = @cod_Estado)
            THROW 50000, 'ERROR. Ya existe', 1

       INSERT INTO informeVENTAS
              SELECT @fechaInforme, @cod_Estado, o.customer_num,
                     COUNT(distinct o.order_num),
                        MAX(o.order_date), COUNT(DISTINCT i.stock_num),
                        SUM(i.unit_price*i.quantity)
         FROM customer c join orders o on c.customer_num = o.customer_num
                                    JOIN items i ON o.order_num = i.order_num
              where c.state = @cod_Estado
              GROUP BY o.customer_num;
```

```
--
END;

-- prueba
declare @fecha datetime
set @fecha = getDate()
EXEC generarInformeVentas @fecha,'CA'

select * from informeVENTAS
```