a. Stored Procedures

Crear la siguiente tabla CustomerStatistics con los siguientes campos customer_num (entero y pk), ordersqty (entero), maxdate (date), uniqueProducts (entero)

Crear un procedimiento 'actualizaEstadisticas' que reciba dos parámetros customer_numDES y customer_numHAS y que en base a los datos de la tabla customer cuyo customer_num estén en en rango pasado por parámetro, inserte (si no existe) o modifique el registro de la tabla CustomerStatistics con la siguiente información:

Ordersqty contedrá la cantidad de órdenes para cada cliente.
Maxdate contedrá la fecha máxima de la última órden puesta por cada cliente.
uniqueProducts contendrá la cantidad única de tipos de productos adquiridos por cada cliente.

```
create  table  CustomerStatistics
(customer_num integer primary key,
ordersqty   integer,       maxdate
datetime,  uniqueManufact integer)

CREATE PROCEDURE actualizaEstadisticas
@customer_numDES INT , @customer_numHAS INT
AS
BEGIN
   DECLARE CustomerCursor CURSOR FOR
   SELECT customer_num from customer WHERE customer_num
            BETWEEN @customer_numDES AND @customer_numHAS

  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;

      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,maxDate=@maxDate,
                    uniqueManufact= @uniqueManufact
             WHERE customer_num = @customer_num;
     FETCH NEXT FROM CustomerCursor INTO @customer_num
  END;
  CLOSE CustomerCursor;
  DEALLOCATE CustomerCursor;
 END
```

```
SELECT * FROM CustomerStatistics

execute actualizaEstadisticas 101,110
```

b. Crear un procedimiento 'migraClientes' que reciba dos parámetros customer_numDES y customer_numHAS y que dependiendo el tipo de cliente y la cantidad de órdenes los inserte en las tablas clientesCalifornia, clientesNoCaBaja, clienteNoCAAlta.

- **El procedimiento deberá migrar de la tabla customer todos los clientes de California a la tabla clientesCalifornia, los clientes que no son de California pero tienen más de 999u$ en OC en clientesNoCaAlta y los clientes que tiene menos de 1000u$ en OC en la tablas clientesNoCaBaja.**
- **Se deberá actualizar un campo status en la tabla customer con valor 'P' Procesado, para todos aquellos clientes migrados.**
- **El procedimiento deberá contemplar toda la migración como un lote, en el caso que ocurra un error, se deberá informar el error ocurrido y abortar y deshacer la operación.**

```
CREATE TABLE [dbo].[clientesCalifornia](
	[customer_num] [smallint] NOT NULL,
	[fname] [varchar](15),
	[lname] [varchar](15),
	[company] [varchar](20),
	[address1] [varchar](20),
	[address2] [varchar](20),
	[city] [varchar](15) ,
	[state] [char](2) ,
	[zipcode] [char](5),
	[phone] [varchar](18)
)

CREATE TABLE [dbo].[clientesNoCaBaja](
	[customer_num] [smallint] NOT NULL,
	[fname] [varchar](15) ,
	[lname] [varchar](15) ,
	[company] [varchar](20),
	[address1] [varchar](20),
	[address2] [varchar](20),
	[city] [varchar](15) ,
	[state] [char](2) ,
	[zipcode] [char](5),
	[phone] [varchar](18)
)

CREATE TABLE [dbo].[clientesNoCaAlta](
	[customer_num] [smallint] NOT NULL,
	[fname] [varchar](15) ,
	[lname] [varchar](15) ,
	[company] [varchar](20),
	[address1] [varchar](20),
	[address2] [varchar](20),
	[city] [varchar](15) ,
	[state] [char](2) ,
	[zipcode] [char](5),
	[phone] [varchar](18)
)
```

```
ALTER TABLE customer ADD status CHAR(1)



CREATE PROCEDURE migraClientes @customer_numDES INT,
                               @customer_numHAS INT
AS
BEGIN
    BEGIN TRY
    DECLARE @customer_num INT,@lname VARCHAR(15),
            @fname VARCHAR(15),@company VARCHAR(20),
            @address1 VARCHAR(20),@address2 VARCHAR(20),
            @city VARCHAR(15),@state CHAR(2),
            @zipcode CHAR(5),@phone VARCHAR(18),
            @status CHAR(1)

  DECLARE CustomerCursor CURSOR FOR
    SELECT customer_num,lname,fname,company,address1,
address2,city,state,zipcode,phone
    FROM customer
    WHERE customer_num
    BETWEEN @customer_numDES AND @customer_numHAS

    OPEN CustomerCursor;
    FETCH NEXT FROM CustomerCursor
               INTO @customer_num,@lname,@fname,@company,
                    @address1,@address2,@city,@state,
                    @zipcode,@phone

    BEGIN TRANSACTION
    WHILE @@FETCH_STATUS = 0
    BEGIN

     IF @state='CA'
                  insert  into  clientesCalifornia
(customer_num,lname,fname,company,
address1,address2,city,state,
zipcode,phone)
        values (@customer_num,@lname,@fname,@company,
                @address1,@address2,@city,@state,
                @zipcode,@phone);
     ELSE
       BEGIN
         IF (SELECT sum(total_price)
               FROM orders o JOIN items i
                 ON (o.order_num = i.order_num)
              WHERE customer_num = @customer_num) > 999

              insert into clientesNoCaAlta
(customer_num,lname,fname,company,
address1,address2,city,state,                          zipcode,phone)
              values (@customer_num,@lname,@fname,@company,
                      @address1,@address2,@city,@state,
                      @zipcode,@phone);
ELSE
              insert into clientesNoCaBaja
(customer_num,lname,fname,company,
address1,address2,city,state,                          zipcode,phone)
              values (@customer_num,@lname,@fname,@company,
```

```
                    @address1,@address2,@city,@state,
                    @zipcode,@phone);
      END

     UPDATE customer SET status= 'P'
     WHERE customer_num= @customer_num


    FETCH NEXT FROM CustomerCursor
              INTO @customer_num,@lname,@fname,@company,
                   @address1,@address2,@city,@state,
                   @zipcode,@phone


    END;
    COMMIT TRANSACTION
    CLOSE CustomerCursor
    DEALLOCATE CustomerCursor
  END TRY
  BEGIN CATCH
      CLOSE CustomerCursor
      DEALLOCATE CustomerCursor
      ROLLBACK TRANSACTION
      DECLARE @errorDescripcion VARCHAR(100)
      SELECT @errorDescripcion = 'Error en Cliente '+CAST(@customer_num AS
CHAR(5))
      THROW 50000, @errorDescripcion, 1
  END CATCH
END;

drop procedure migraClientes


--Pruebas
SELECT count(*) FROM clientesCalifornia select
count(*) from customer


exec migraClientes 100,126

select count(*) from customer where customer_num between 100 and 126
select count(*) from clientesCalifornia select count(*) from
clientesNoCaAlta select count(*) from clientesNoCaBaja select
count(*) from customer  where customer_num between 100 and 126 and
status='P'
 delete from
clientesCalifornia delete from
clientesNoCaAlta delete from
clientesNoCaBaja
```

c. Crear un procedimiento 'actualizaPrecios' que reciba como parámetros manu_codeDES, manu_codeHAS y porcActualizacion que dependiendo del tipo de cliente y la cantidad de órdenes genere las siguientes tablas listaPrecioMayor y listaPreciosMenor. Ambas tienen las misma estructura que la tabla Productos.

- **El procedimiento deberá tomar de la tabla Productos todos los productos que correspondan al rango de fabricantes asignados por parámetro.**
  Por cada producto del fabricante se evaluará la cantidad (quantity) comprada.

Si la misma es mayor o igual a 500 se grabará el producto en la tabla listaPrecioMayor y el unit_price deberá ser actualizado con (unit_price * (porcActualización *0,80)),
Si la cantidad comprada del producto es menor a 500 se actualizará (o insertará) en la tabla listaPrecioMenor y el unit_price se actualizará con (unit_price * porcActualizacion)

- **Asimismo, se deberá actualizar un campo *status* de la tabla stock con valor 'A' Actualizado, para todos aquellos productos con cambio de precio actualizado.**
- **El procedimiento deberá contemplar todas las operaciones de cada fabricante como un lote, en el caso que ocurra un error, se deberá informar el error ocurrido y deshacer la operación de ese fabricante.**

```
CREATE TABLE [dbo].[listaPrecioMayor](
       [stock_num] [smallint] NOT NULL,
       [manu_code] [char](3) NOT NULL,
       [unit_price] [decimal](6, 2) NULL,
       [unit_code] smallint
);

CREATE TABLE [dbo].[listaPrecioMenor](
       [stock_num] [smallint] NOT NULL,
       [manu_code] [char](3) NOT NULL,
       [unit_price] [decimal](6, 2) NULL,
       [unit_code] smallint
);

ALTER TABLE products ADD status char(1);

ALTER PROCEDURE actualizaPrecios @manu_codeDES CHAR(3), @manu_codeHAS CHAR(3),
                                 @porcActualizacion decimal (5,3)
AS
BEGIN
    DECLARE @stock_num INT, @manu_code CHAR(3), @unit_price DECIMAL(6,2),
            @unit_code smallint, @manu_codeAux CHAR(3)

    DECLARE StockCursor CURSOR FOR
                SELECT p.stock_num, manu_code, unit_price, unit_code
                  from products p
                 WHERE manu_code  BETWEEN @manu_codeDES AND @manu_codeHAS
              ORDER BY manu_code, p.stock_num

    OPEN StockCursor;
    FETCH NEXT FROM StockCursor INTO @stock_num, @manu_code,
                                     @unit_price,@unit_code
    set @manu_codeAux = @manu_code

    WHILE @@FETCH_STATUS = 0
    BEGIN
        BEGIN TRY
        BEGIN TRANSACTION
              IF ( SELECT sum(quantity) FROM items
              WHERE manu_code = @manu_code AND stock_num=@stock_num) >= 500
                insert into listaPrecioMayor
                 values (@stock_num, @manu_code,
                     @unit_price * (1 + @porcActualizacion * 0.80), @unit_code);
              ELSE
                insert into listaPrecioMenor
                 values (@stock_num,@manu_code,
                           @unit_price * (1 + @porcActualizacion), @unit_code);

              UPDATE products SET status= 'A'
```

```
            WHERE manu_code= @manu_code AND stock_num= @stock_num;

        FETCH NEXT FROM StockCursor INTO @stock_num, @manu_code,
                                         @unit_price, @unit_code

        IF @manu_code != @manu_codeAux
        BEGIN
              COMMIT TRANSACTION
              SET @manu_codeAux = @manu_code
        END
        END TRY
        BEGIN CATCH
             ROLLBACK TRANSACTION
             DECLARE @errorDescripcion VARCHAR(100)
             SELECT @errorDescripcion = 'Error en Fabricante '+@manu_code
             RAISERROR(@errorDescripcion,14,1)
        END CATCH
     END;
     CLOSE StockCursor
     DEALLOCATE StockCursor
END;

-- Pruebas
delete from listaPrecioMayor
delete from listaPrecioMenor
update products set status=''

insert into items values (2,1001,1,'HRO',1000,250.00)

exec actualizaPrecios 'HRO','HRO',0.10

select * from listaPrecioMayor
select * from products where stock_num=1 and manu_code='HRO'

select * from listaPrecioMenor where stock_num=2 select * from
products  where  stock_num=2  and  manu_code='HRO'  select
count(*) from listaPrecioMenor

select * from products where manu_code = 'ANZ'

exec actualizaPrecios 'ANZ','ANZ',0.05

select * from products where manu_code = 'ANZ'
```