Ejercitación 2:

1. Crear tabla ordenes\_cli (row\_id, nom\_cli, nro\_pedido, nuevo\_estado, resultado). Debe seleccionar los tipos de datos de acuerdo a su criterio (leer todo el ejercicio primero).
2. Actualizar la columna accnt\_id de la tabla s\_order y colocar, para todos los registros, el valor: '0-5200'
3. Realizar un Insert sobre la tabla ordenes\_cli con los siguientes valores:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| nro\_pedido | 11111 | nom\_cli | Siebel Administration | nuevo\_estado | Open |
| nro\_pedido | 11112 | nom\_cli | Siebel Power | nuevo\_estado | Open |
| nro\_pedido | 11113 | nom\_cli | Siebel Administration | nuevo\_estado | Abierto |
| nro\_pedido | 11119 | nom\_cli | Siebel Administration | nuevo\_estado | Billed |
| nro\_pedido | 11120 | nom\_cli | Siebel Administration | nuevo\_estado | Billed |
| nro\_pedido | 11114 | nom\_cli | Siebel | nuevo\_estado | Billed |
| nro\_pedido | 11115 | nom\_cli | SADMIN | nuevo\_estado | Billed |
| nro\_pedido | 11116 | nom\_cli | Siebel Administration | nuevo\_estado | Billed |

La columna resultado se insertará en null para todos los registros, el row\_id deberá ser la primary key de cada registro, y un valor numérico único para cada caso.

1. Crear un package que posea 4 procedimientos:
2. Función -> valida\_existencia\_pedido, recibe el id de la tabla ordenes\_cli y verifica en la tabla s\_order si existe o no el nro\_pedido en la columna order\_num de dicha tabla. En caso de no existir, deberá devolver el valor N y, en caso de hacerlo, devolver el valor Y.
3. Función -> valida\_existencia\_cliente, recibe el id de la tabla ordenes\_cli y verifica en la tabla s\_org\_ext si la columna nom\_cli coincide con la columna name de dicha tabla. En caso de coincidir, se deberá buscar en la tabla de pedidos (s\_order) por la columna nro\_pedido, que la columna accnt\_id sea = al row\_id del registro encontrado en la s\_org\_ext. En caso de no cumplirse alguna de las 2 condiciones, la función debe retornar el valor N. En caso de cumplirse ambas, retornar el valor Y.
4. Función -> valida\_estado, recibe el id de la tabla ordenes\_cli y verifica en la tabla s\_lst\_of\_val que el valor de la columna nuevo\_estado coincida para el TYPE FS\_ORDER\_STATUS y activo. (Para ayudarse con los campos físicos de la tabla, pueden hacer un about view de la vista de LOVs). En caso de existir el estado, devolver el valor Y. En caso de no hacerlo, retornar N.
5. Función -> actualiza\_orden, recorre todos los registros de la tabla ordenes\_cli y, para cada uno de ellos, deberá ejecutar las fuciones de los puntos a, b y c. En caso de que la función A retorne el valor N, se deberá actualizar la tabla ordenes\_cli (columna resultado) con la leyenda: ‘No existe pedido en Siebel.’

En caso de que la función B retorne el valor N, se deberá actualizar la tabla ordenes\_cli (columna resultado) con la leyenda: ‘No existe el cliente en Siebel.’

En caso de que la función C retorne el valor N, se deberá actualizar la tabla ordenes\_cli (columna resultado) con la leyenda: ‘El estado a actualizar no es válido en Siebel.’

En caso de que todas las validaciones funcionen OK, se deberá actualizar la columna STATUS\_CD de la tabla S\_ORDER por nro\_pedido, en base al valor que posea la columna nuevo\_estado y, se deberá actualizar la columna resultado con la leyenda: ‘No hubo errores’.

NOTA: Puede verificar en Google la sintaxis de creación de un pkg.

BONUS:

1. Crear una tabla auditoria\_pedidos (row\_id, id\_pedido, usuario). Debe seleccionar los tipos de datos de acuerdo a su criterio.
2. Cuando se realice una actualización de la columna STATUS\_CD de la tabla S\_ORDER al valor “Open”, se deberá crear inmediatamente un nuevo registro en la tabla auditoria\_pedidos con el row\_id del pedido actualizado, y el usuario que realizó la modificación.