* ***Escriba la sintaxis, en formato BNF, de las sentencias del lenguaje SQLXX\_β (donde XX es el número de grupo), que cumpla con las siguientes consignas:***

**Sentencia: select**

**Alternativa 1:**

**select \* from** Tabla [**where** Condición] [**order by** ListaColumnasOrd]

donde:

* Tabla: es el nombre de la tabla.
* Condición: especifica la condición de filtro y puede ser una condición binaria simple con operador lógico (>, >=, >, <=, ==,!=), donde ambos operandos son nombres de columnas, o más de una condición binaria simple conectadas con conectores lógicos **and** y **or**.
* [**where** Condición]:Los corchetes indican que la condición es opcional.
* *ListaColumnasOrd* es una lista de uno o más nombres de columnas separados por coma, con su ordenamiento. Debe especificarse para cada columna su tipo de ordenamiento **asc** o **desc**: Ejemplo: id1 **asc**, id2 **desc**, id3 **desc**.
* [**order by** ListaColumnasOrd]: Los corchetes indican que el ordenamiento de columnas es opcional.
* Las palabras reservadas **select, from, where, order by, and, or, asc** y **desc** son terminales.
* Los nombres de columnas y nombre de tabla serán terminales **id.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

**Alternativa 2:**

**select** [**distinct**] ListaColumnas **from** ListaTablas **where** Condición [**order by** ListaColumnasOrd]

donde:

* [**distinct**] Los corchetes indican que la palabra reservada **distinct** es opcional.
* ListaColumnas: es una lista de una o más nombres de columnas de tablas separadas por comas.
* ListaTablas: es una lista de una o más nombres de tablas, separadas por coma.
* Condición especifica la condición de filtro y puede ser una condición binaria simple con operador lógico (>, >=, >, <=, ==,!=), donde ambos operandos son nombres de columnas, o más de una condición binaria simple conectadas con conectores lógicos **and** y **or**.
* *ListaColumnasOrd* es una lista de uno o más nombres de columnas separados por comas, con ordenamiento. Debe especificarse para cada columna el tipo de ordenamiento **asc** o **desc**: Ejemplo: id1 **asc**, id2 **desc**, id3 **desc**
* [**order by** ListaColumnasOrd]: Los corchetes indican que el ordenamiento de columnas es opcional.
* Las palabras reservadas **select, from, where, order by, distinct, and, or, asc y desc** son terminales.
* Los nombres de columnas y tablas serán terminales **id.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

**Alternativa 3:**

**Uso del operador unión: Se pueden unir dos sentencias select como las que vimos en las alternativas 1 y 2 con la siguiente sintaxis:**

sentencia select **union** sentencia select

**Sentencia: update**

**update** Tabla **set** Columna1=valor1, Columna2=valor2,.. Columnan=valorn [**where** Condición]

donde:

* Columnai: es el nombre de una columna. Puede ser una columna o más de una, cada una con la asignación de su valor.
* Valori: es una constante numérica o alfanumérica.
* Tabla: es el nombre de una tabla.
* [**where** Condición] especifica la condición de filtro y puede ser una condición binaria simple con operador lógico (>, >=, >, <=, ==,!=), donde ambos operandos son nombres de columnas, o más de una condición binaria simple conectadas con conectores lógicos **and** y **or.** Los corchetes indican que la condición es opcional.
* Las palabras reservadas **set, and or y where** son terminales.
* Los nombres de columnas y tablas serán terminales **id.**
* Los valores constantes de las columnas serán terminales **cte.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

**Sentencia: delete**

**delete from**  Tabla [**where** Condición]

* Tabla: es el nombre de una tabla.
* [where Condición] especifica la condición de filtro y puede ser una condición binaria simple con operador lógico (>, >=, >, <=, ==,!=), donde ambos operandos son nombres de columnas, o más de una condición binaria simple conectadas con conectores lógicos **and y or**. Los corchetes indican que la condición es opcional.
* Las palabras reservadas **delete, from, where, and y or** son terminales.
* Los nombres de tablas serán terminales **id.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

**Sentencia: insert**

**insert into** Tabla (Campo1, Campo2, …Campon) **values** (Valor1, Valor2, ..Valorn)

donde

* Campoi: es el nombre del campo de una tabla. Puede ser uno o más campos.
* Valori: es una constante numérica o alfanumérica. La cantidad de valores debe ser igual a la cantidad de campos.
* Tabla: es el nombre de una tabla.
* Las palabras reservadas **insert, into, values** son terminales.
* Los nombres de campos y tablas serán terminales **id.**
* Los valores de los campos serán terminales **cte.**
* Todos los símbolos unarios son parte del lenguaje al que pertenece la sentencia.

En un programa del lenguaje ***SQLXX\_β***  puede haber una o más sentencias de las definidas, separadas por punto y coma.

* ***Escriba un programa en el lenguaje SQLXX\_β*** : escriba un programa en el lenguaje SQLXX\_ ***β*** que cumpla con la sintaxis del mismo.
* ***Implemente en Jflap***: Diseñar un Parser LL y otro parser LR de las reglas gramaticales de dos sentencias del lenguaje de programación, en el aplicativo mencionado. Tenga en cuenta que Jflap no acepta notación BNF. Modifique la notación (sólo los elementos no terminales) para que pueda implementar los parsers.
* ***Entregables:*** Documento digital que contenga:
* Las reglas de la gramática del lenguaje de programación,
* El programa escrito en el lenguaje,
* Las gramáticas modificadas de las sentencias elegidas para armar los parsers LL y LR,
* Parser LL y LR de las gramáticas de las sentencias elegidas (archivo de extensión .flap de los parsers diseñados).