問11.

現在のテーブルに一番右に、「star」というカラムを追加して参照してください。  
　なお、「star」カラムには、総合得点（sum）が370点以上の人に、「☆」の値が入るカラムとします。という問題です．ただし，回答方法にはjoin句を使用との指定があります，

この問題の考え方ですが，

JOIN句とあるので，複数のテーブルを扱うということになります．

目標は，もとのテーブルには存在しない，「☆」の値が入るカラムを追加することなので，

もとのテーブルに加えて，結合相手のテーブルに，「☆」の値が入るカラムもたせることを考えます．

また，今回 370点以上の人にしか印をつけないので．もとのテーブルから，

370点以上の人だけを抽出し，その抽出したデータにのみ，「☆」を入れれば，条件にあう人のみ，印を付与できます．

これを図でみると，

まずもとのテーブルをTable1とします．また，結合先のテーブルをTable2としたとき，

Table2は，もとのテーブルから，総得点が370点以上の人だけを抽出したものとします．

そして，このテーブルに新たな列として， 文字型の「☆」を追加します．

このテーブル２は，総得点が370点以上の人のみなので，Table1と，全員に固有の値をもつカラムである [temp\_no]で結合すれば，現在のテーブルの総得点が370点以上のみに「☆」の値が入るカラムを付与できます．

コードにするとこのようになっています．

ポイントは2つのテーブルを結合しているということ，結合先のテーブルをこの部分で指定していますが，その作成も，

一つの構文の中で，もとのテーブルから作成しているということです．

まずは結合先テーブル先ほどでいう，table2を作成している部分から見ていきましょう．

結合先のテーブルには総得点が370点以上の人だけを抽出したいので，where句で総得点を表すからむ[sum]が370以上の人を抽出しています．もとのテーブルと結合する必要があるので，結合のキーとなる列今回であれば， [temp\_no]もselect列に指定します．そして，回答では[star]という列名にしていますが，星形の文字を持つカラムはこのようにselect句に直接指定して追加することができます．ここまでがtable2の作成です．

結合先テーブルが作成できたので，最後にもとのテーブルと [temp\_no]で結合します．もとのテーブルの形をのこしたまま，結合先テーブルの列を追加する必要があるので

今回はLEFT OUTER JOINを用います．キーは[temp\_no]です．

こうすることで， もとのテーブルの情報をすべて出力します．

INNER JOINを使うと，結合した結果，総得点[sum]が370以上の人しか抽出できなくなるので，注意してください．

【TIPS】

ここで，テーブル間の結合に用いる，INNER JOIN とOUTER JOINについて整理しておきましょう．

まず，INNER JOINからみていきます．

こちらがJOINの基本構文で，FROM句の後にJOIN句を記述します．

この際，FROM句のあとにくるテーブルを左側のテーブル，JOINの後にくるテーブルを右側のテーブルとすると，INNER JOINでは結合に用いた列が，テーブル間で一致するレコードのみを取り出します．

図で見てみると，このように例えばこの共通のｘ１という項目を持つテーブルをｘ１をキーとして結合したとき，両方で共通する，A,Bという値をもつレコードのみが出力されます．

結合の相手がいない場合は，そのデータは出力されずに，もとのデータ件数の不一致が生じる可能性があるので，注意が必要です．

結合相手がいない場合にも，もとデータを残す場合は次ページ以降のOUTER JOINを使います．

ここでは

LEFT OUTER JOINとRIGHT OUTER JOINがありますがこれらの違いは

先程の左側と右側のテーブルのうち，どちらをすべてのこしたまま，出力するかの違いがあります．

図はLEFT OUTER JOINの例ですが，２つのテーブルを共通のｘ１という列で結合した場合，左側のテーブルのCという値を持つレコードは，右側に結合相手がいません．INNER JOINでは出力されなかったこのレコードですが，LEFTOUTER JOINでは出力されます．このとき，対応するレコードがないばあいはNULLになって帰ってきます．

問11を問題の指定ではJOINで解いていましたがCASE式というものを使えばより簡単な構文で回答できるのでこちらも紹介します．

CASE式を用いると条件分岐で値を変換することがきます．

基本構文はこちらにあるようにCASE WHEN と書いてあげて条件を指定します

その後条件が真のときに返す値をTHENの後に指定します．最後にこの条件に合致しないときに返す値をELSEの後に記述します．

この問題の場合は総得点が370以上であれば星マーク，それ以外ではNULLを返せばよいので，CASEWHENのあとの条件に[総得点を表すカラムsum]>=370と記述し，これを満たす場合に，返す値として，星マークをTHENの後に指定しています． 370点未満ではNULLを返せばよいので，ELSEのあとにNULLを指定すれば先程のJOIN構文と同じ回答になる．

このようにCASE式を使えば，場合よっては簡単に記述できることもあるので

ぜひ覚えておいてください．

問１２．クラスごとにあいうえお順で出席番号を発番し、Class\_Noに値を入れてください

この問題は，実際にテーブルに値を追加する練習を行います．

実施する事項としては，「クラスごとに出席番号を発番」するというところと，その値をテーブルに「追加する」というところです．

この値を追加する際はUPDATE句を使います．

後でも述べますが，UPDATE句を使うとテーブルが更新されて元の状態に戻すことができないので，トランザクションという機能を使いなが正しく反映されるか確認しながら実行していきます．

まずは更新の前に「クラスごとにあいうえお順で出席番号を発番」という部分の実装です．

出席番号のような連番を発番する際はウインドウ関数というものが使えます．

この部分．今回であれば，ROW\_NUMBRというものがウインドウ関数に当たります．

今回はクラスごとにあいうえお順で出席番号を発番するので，

OVER句にて，PARTITION BY も後に， [class]を指定することで，クラスごとに1から連番を発番します．

また，あいうえお順というものは項目[reading]を昇順に並べたうえで連番を付与していけば実現できるので． ORDERBY句で[reading]を記述します．このようにすることで，PARTITION BYでクラスごと，ORDERBYであいうえ順という指定の下，ROW\_NUMBERで連番を発番するという構造です．

【TIPS】

ここで，今のROW＿NUMBRT関数の挙動についてもう少し丁寧に見ていきます．

構文はこの形で，先ほどあったPARTITION BYは省略可能です．

ここで，ORDER BYはこれまでの口座ででてきたように列指定することでその列で並び替えるというオプションです．デフォルトでは昇順となることに注意してください，

そしてPARTITION BYという部分ですが，PARTITION BYで列名を指定することでイメージとしてはグループ倍のような形で，対象の列の値ごとにテーブルを分割し，

それぞれの部分でROW＿NUMBRT，連番を発番するという指定が可能です．

例も書いていますが，例えば先ほどと同じ列を指定し，PARTITION BYを指定しない場合は

クラス関係なく単純に項目[reading]で昇順に連番が振られます．

PARTITION BYでclassを指定した場合，クラスごとに項目 [reading]で昇順に並び替えられ，一から連番が振られる．

P１２

UPDATE

ここまでで，ウィンドウ関数でクラスごとにあいうえお順で出席番号を発番しという部分までは実行できたので

最後にテーブルに値を追加していきます．値の追加というよりも更新ですね．

値の更新にはUPDATE句を使います．

構文としてはUPDATEの後に，対象のテーブル名と書き

値を更新したい列名をset句の後に記述します．

今回は，新たに作成したクラスごとの連番で対象の列[class\_no]を更新するが

その際もとのテーブルと，JOIN句で結合することで，値を一括更新しています．

そのために，全生徒に固有の番号である[temp\_no]を，先ほど作成した，

「クラスごとにあいうえお順で出席番号を発番する構文でも出力し，結合条件にしています．

最後に，先程の構文を実行すれば実はそれでテーブルの更新がなされます．

というのも，SQLserverは接続した時点から「一つのSQL分を一つのトランザクションとみなす」というルールがデフォルトで適用されているためです．

なので，UPDATE時にあやまった処理を記述してしまっていたとしても

それがそのまま反映されてしまいます可能性があるという点に注意してください．

テーブルの更新や値の削除といった操作時にはトランザクションという機能用いられます．

トランザクションを用いれば，データの更新処理の確定や取り消しなどを管理することができます．

SQLserverでトランザクションを明示的に開始するには，

BEGIN TRANSACTION；

とまず記述します．

するとこれ以降の処理はトランザクションの処理の確定を意味するCOMMITというコマンドが使用されてまで，トランザクション内の処理は確定されません．

今回であれば．

COMMIT;を実行するまでは実際のテーブルに更新の処理が反映されることはないので，

トランザクション内で，UPDATEした結果を確認し，問題なければCOMMITすることで

本当に更新して問題ないかデータをみながら確認できます．

もし，処理を確定させたくない場合は，ROOLBACKコマンドを実行することで

このトランザクション開始前の状態に戻すことができます．

このトランザクションの考え方というものはぜひ覚えておいてください，