Les requêtes SQL

1. SELECT
   * 1. DISTINCT

TYPES: BEGIN OF ty\_driver\_car,  
         car\_brand TYPE wrf\_brand\_descr,  
         car\_model TYPE vlc\_maktx,  
       END OF ty\_driver\_car.  
  
DATA: t\_driver\_car TYPE TABLE OF ty\_driver\_car,  
      s\_driver\_car TYPE ty\_driver\_car.  
  
SELECT DISTINCT car\_brand, car\_model  
    FROM zdriver\_car  
    INTO TABLE @t\_driver\_car.  
  
LOOP AT t\_driver\_car INTO s\_driver\_car.  
  WRITE:/ s\_driver\_car-car\_brand, s\_driver\_car-car\_model.  
ENDLOOP.

* + 1. Col AS alias

TYPES: BEGIN OF ty\_driver\_car,  
         marque TYPE wrf\_brand\_descr,  
         model  TYPE vlc\_maktx,  
       END OF ty\_driver\_car.  
  
DATA: t\_driver\_car TYPE TABLE OF ty\_driver\_car,  
      s\_driver\_car TYPE ty\_driver\_car.  
  
SELECT DISTINCT car\_brand AS marque,  
                car\_model AS model  
    FROM zdriver\_car  
    INTO TABLE @t\_driver\_car.  
  
LOOP AT t\_driver\_car INTO s\_driver\_car.  
  WRITE:/ s\_driver\_car-marque, s\_driver\_car-model.  
ENDLOOP.

* + 1. Les agrégats

TYPES: BEGIN OF ty\_driver\_infos,  
         id    TYPE zdriver\_id,  
         total TYPE ztravel\_toll,  
         moy   TYPE ztravel\_toll,  
         max   TYPE n1dauer,  
         min   TYPE n1dauer,  
       END OF ty\_driver\_infos.  
  
DATA: t\_driver\_infos TYPE TABLE OF ty\_driver\_infos,  
      s\_driver\_infos TYPE ty\_driver\_infos.  
  
  
SELECT id\_driver       AS id,  
       SUM( toll )     AS total,  
       AVG( toll )     AS moy,  
       MAX( duration ) AS max,  
       MIN( duration ) AS min  
  FROM ztravel  
  INTO TABLE @t\_driver\_infos  
  GROUP BY id\_driver.  
  
LOOP AT t\_driver\_infos INTO s\_driver\_infos.  
  WRITE:/ s\_driver\_infos-id,  s\_driver\_infos-total,  
          s\_driver\_infos-moy, s\_driver\_infos-max,  
          s\_driver\_infos-min.  
ENDLOOP.

* + 1. Les expressions SQL
* Expressions arithmétiques

TYPES: BEGIN OF ty\_cost,  
         city\_from  TYPE s\_from\_cit,  
         city\_to    TYPE s\_to\_city,  
         total\_traj TYPE p DECIMALS 2,  
       END OF ty\_cost.  
  
DATA: t\_cost TYPE TABLE OF ty\_cost,  
      s\_cost TYPE ty\_cost.  
  
  
SELECT city\_from,  
       city\_to,  
       ( toll + gasol ) AS total\_traj  
  FROM ztravel  
  INTO TABLE @t\_cost.  
  
LOOP AT t\_cost INTO s\_cost.  
  WRITE:/ s\_cost-city\_from, s\_cost-city\_to, s\_cost-total\_traj.  
ENDLOOP.

* Les fonctions arithmétiques

SELECT toll,  
       duration,  
       gasol,  
       ABS( toll - gasol ) AS abs,    "Valeur absolue  
       CEIL( gasol )       AS ceil,   "Arrondi inférieur  
       FLOOR( gasol )      AS floor,  "Arrondi supérieur  
       DIV( duration,60 )  AS div,    "Division  
       MOD( duration,60 )  AS mod     "Reste division  
  FROM ztravel  
  INTO TABLE @DATA(t\_arith).  
  
  
DATA: s\_arith LIKE LINE OF t\_arith.  
  
  
LOOP AT t\_arith INTO s\_arith.  
  WRITE:/ s\_arith-toll, s\_arith-duration, s\_arith-gasol,  
          s\_arith-abs,  s\_arith-ceil,     s\_arith-floor,  
          s\_arith-div,  s\_arith-mod.  
ENDLOOP.

* La fonction de conversion

CONSTANTS: c\_60(2) TYPE i VALUE '60'.  
  
SELECT city\_from,  
       city\_to,  
       kms,  
       duration,  
       CAST( kms AS FLTP ) \* CAST( @c\_60 AS FLTP ) / CAST( duration AS FLTP ) AS vitesse  
  FROM ztravel  
  INTO TABLE @DATA(t\_cast).  
  
  
DATA: s\_cast LIKE LINE OF t\_cast.  
  
LOOP AT t\_cast INTO s\_cast.  
  WRITE:/ s\_cast-city\_from, s\_cast-city\_to, s\_cast-kms,  
          s\_cast-duration,  s\_cast-vitesse.  
ENDLOOP.

DATA:      v\_vit   TYPE p DECIMALS 2.  
CONSTANTS: c\_60(2) TYPE i VALUE '60'.  
  
  
SELECT city\_from,  
       city\_to,  
       kms,  
       duration,  
       CAST( kms AS FLTP ) \* CAST( @c\_60 AS FLTP ) / CAST( duration AS FLTP ) AS vitesse  
  FROM ztravel  
  INTO TABLE @DATA(t\_cast).  
  
  
DATA: s\_cast LIKE LINE OF t\_cast.  
  
LOOP AT t\_cast INTO s\_cast.  
  CLEAR v\_vit.                "Nettoyage de la variable  
  v\_vit = s\_cast-vitesse.     "Transférer la valeur de VITESSE

"dans la variable V\_VIT  
  
  WRITE:/ s\_cast-city\_from, s\_cast-city\_to, s\_cast-kms,  
          s\_cast-duration,  v\_vit.  
  
ENDLOOP.

* Le Case
* Le CASE simple

SELECT name,  
       surname,  
       city,  
       country,  
       CASE country  
         WHEN 'FR' THEN 'FRANCE'  
         WHEN 'ES' THEN 'ESPAGNE'  
       END AS pays  
  FROM zdriver\_car  
  INTO TABLE @DATA(t\_drivers).  
  
  
DATA s\_drivers LIKE LINE OF t\_drivers.  
  
LOOP AT t\_drivers INTO s\_drivers.  
  WRITE:/ s\_drivers-name,    s\_drivers-surname, s\_drivers-city,  
          s\_drivers-country, s\_drivers-pays.  
ENDLOOP.

* Le CASE complexe

SELECT city\_from && ' - ' && city\_to AS traject,  
       kms,  
       CASE WHEN kms <= 200              THEN 'Trajet court'  
            WHEN kms BETWEEN 200 AND 300 THEN 'Trajet moyen'  
            WHEN kms > 300               THEN 'Trajet long'  
            ELSE 'Non défini'  
       END AS type\_traj  
  FROM ztravel  
  INTO TABLE @DATA(t\_traject).  
  
DATA s\_traject LIKE LINE OF t\_traject.  
  
LOOP AT t\_traject INTO s\_traject.  
  WRITE:/ s\_traject-traject, s\_traject-kms, s\_traject-type\_traj.  
ENDLOOP.

* 1. FROM

. . .

SELECT zcar\_brand\_mod~brand,  
       zcar\_brand\_mod~model,  
       zcar\_brand\_mod~model\_year,  
       zdriver\_car~surname,  
       zdriver\_car~name  
  FROM zcar\_brand\_mod INNER JOIN zdriver\_car  
    ON zcar\_brand\_mod~brand       = zdriver\_car~car\_brand  
    AND zcar\_brand\_mod~model      = zdriver\_car~car\_model  
    AND zcar\_brand\_mod~model\_year = zdriver\_car~car\_year  
  INTO TABLE @DATA(t\_car).  
  
  
DATA s\_car LIKE LINE OF t\_car.  
  
LOOP AT t\_car INTO s\_car.  
  WRITE:/ s\_car-brand,   s\_car-model, s\_car-model\_year,  
          s\_car-surname, s\_car-name.  
ENDLOOP.

SELECT b~brand,  
       b~model,  
       b~model\_year,  
       d~surname,  
       d~name  
  FROM zcar\_brand\_mod AS b INNER JOIN zdriver\_car AS d  
    ON  b~brand      = d~car\_brand  
    AND b~model      = d~car\_model  
    AND b~model\_year = d~car\_year  
  INTO TABLE @DATA(t\_car).  
  
  
DATA s\_car LIKE LINE OF t\_car.  
  
LOOP AT t\_car INTO s\_car.  
  WRITE:/ s\_car-brand,   s\_car-model, s\_car-model\_year,  
          s\_car-surname, s\_car-name.  
ENDLOOP.

SELECT b~brand,  
       b~model,  
       b~model\_year,  
       d~surname,  
       d~name  
  FROM zcar\_brand\_mod AS b LEFT JOIN zdriver\_car AS d  
    ON  b~brand      = d~car\_brand  
    AND b~model      = d~car\_model  
    AND b~model\_year = d~car\_year  
  INTO TABLE @DATA(t\_car\_left).

DATA s\_car\_left LIKE LINE OF t\_car\_left.  
  
LOOP AT t\_car\_left INTO s\_car\_left.  
  WRITE:/ s\_car\_left-brand,    s\_car\_left-model,

s\_car\_left-model\_year, s\_car\_left-surname,

s\_car\_left-name.  
ENDLOOP.

SELECT d~surname,  
       d~name,  
       b~brand,  
       b~model,  
       b~model\_year  
  FROM zdriver\_car AS d RIGHT JOIN zcar\_brand\_mod AS b  
    ON  d~car\_brand = b~brand  
    AND d~car\_model = b~model  
    AND d~car\_year  = b~model\_year  
  INTO TABLE @DATA(t\_car\_right).  
  
  
DATA s\_car\_right LIKE LINE OF t\_car\_right.

LOOP AT t\_car\_right INTO s\_car\_right.

WRITE:/ s\_car\_right-brand, s\_car\_right-model,

s\_car\_right-model\_year, s\_car\_right-surname,

s\_car\_right-name.

ENDLOOP.

SELECT b~brand,  
       b~model,  
       b~model\_year,  
       COALESCE( d~surname, 'X' ) AS surname,  
       COALESCE( d~name, 'X' )    AS name  
  FROM zcar\_brand\_mod AS b LEFT JOIN zdriver\_car AS d  
    ON  b~brand      = d~car\_brand  
    AND b~model      = d~car\_model  
    AND b~model\_year = d~car\_year  
  INTO TABLE @DATA(t\_car\_left).  
  
  
DATA s\_car\_left LIKE LINE OF t\_car\_left.

LOOP AT t\_car\_left INTO s\_car\_left.

WRITE:/ s\_car\_left-brand, s\_car\_left-model,

s\_car\_left-model\_year, s\_car\_left-surname,

s\_car\_left-name.

ENDLOOP.

* 1. INTO

DATA: t\_brand\_mod TYPE TABLE OF zcar\_brand\_mod.  
FIELD-SYMBOLS: <fs\_brand\_mod> TYPE zcar\_brand\_mod.  
  
SELECT model\_year,  
       brand,  
       model  
  FROM zcar\_brand\_mod  
  INTO TABLE @t\_brand\_mod.  
  
LOOP AT t\_brand\_mod ASSIGNING <fs\_brand\_mod>.  
  WRITE:/ <fs\_brand\_mod>-model\_year, <fs\_brand\_mod>-brand,  
          <fs\_brand\_mod>-model.  
ENDLOOP.

DATA: t\_brand\_mod TYPE TABLE OF zcar\_brand\_mod.  
FIELD-SYMBOLS: <fs\_brand\_mod> TYPE zcar\_brand\_mod.  
  
SELECT model\_year,  
       brand,  
       model  
  FROM zcar\_brand\_mod  
  INTO CORRESPONDING FIELDS OF TABLE @t\_brand\_mod.  
  
LOOP AT t\_brand\_mod ASSIGNING <fs\_brand\_mod>.  
  WRITE:/ <fs\_brand\_mod>-model\_year, <fs\_brand\_mod>-brand,  
          <fs\_brand\_mod>-model.  
ENDLOOP.

* APPENDING

SELECT surname,  
       name,  
       date\_birth  
  FROM zpassenger  
  INTO TABLE @DATA(t\_passenger).  
  
SELECT surname,  
       name,  
       date\_birth  
  FROM zpassenger  
  INTO TABLE @t\_passenger.  
  
  
DATA s\_passenger LIKE LINE OF t\_passenger.  
  
LOOP AT t\_passenger INTO s\_passenger.  
  WRITE:/ s\_passenger-surname, s\_passenger-name,  
          s\_passenger-date\_birth.  
ENDLOOP.

SELECT surname,  
       name,  
       date\_birth  
  FROM zpassenger  
  INTO TABLE @DATA(t\_passenger).  
  
SELECT surname,  
       name,  
       date\_birth  
  FROM zpassenger  
  APPENDING TABLE @t\_passenger.  
  
  
DATA s\_passenger LIKE LINE OF t\_passenger.  
  
LOOP AT t\_passenger INTO s\_passenger.  
  WRITE:/ s\_passenger-surname, s\_passenger-name,  
          s\_passenger-date\_birth.  
ENDLOOP.

DATA: t\_passenger TYPE TABLE OF zpassenger,  
      s\_passenger TYPE zpassenger.  
  
SELECT surname,  
       name,  
       date\_birth  
  FROM zpassenger  
  INTO CORRESPONDING FIELDS OF TABLE @t\_passenger.  
  
SELECT surname,  
       name,  
       date\_birth  
  FROM zpassenger  
  APPENDING CORRESPONDING FIELDS OF TABLE @t\_passenger.

* 1. WHERE
* Des opérateurs de comparaison

.

SELECT brand,  
       model,  
       model\_year  
  FROM zcar\_brand\_mod  
  INTO TABLE @DATA(t\_car\_mod)  
  WHERE ( brand    =  'PEUGEOT' OR brand = 'VOLKSWAGEN' )  
    AND model\_year >= '2015'.  
  
DATA s\_car\_mod LIKE LINE OF t\_car\_mod.  
  
LOOP AT t\_car\_mod INTO s\_car\_mod.  
 WRITE:/ s\_car\_mod-brand, s\_car\_mod-model, s\_car\_mod-model\_year.  
ENDLOOP.

SELECT city\_from,  
       country\_from,  
       city\_to,  
       country\_to,  
       kms  
  FROM ztravel  
  INTO TABLE @DATA(t\_travel)  
  WHERE kms < ( SELECT max( kms ) FROM ztravel ).  
  
  
DATA s\_travel LIKE LINE OF t\_travel.  
  
LOOP AT t\_travel INTO s\_travel.  
 WRITE:/ s\_travel-city\_from, s\_travel-country\_from,  
 s\_travel-city\_to, s\_travel-country\_to.  
ENDLOOP.

SELECT city\_from,  
       country\_from,  
       city\_to,  
       country\_to,  
       kms  
  FROM ztravel  
  INTO TABLE @DATA(t\_travel)  
  WHERE kms <= ALL ( SELECT kms FROM ztravel ).  
  
  
DATA s\_travel LIKE LINE OF t\_travel.  
  
LOOP AT t\_travel INTO s\_travel.  
 WRITE:/ s\_travel-city\_from, s\_travel-country\_from,  
 s\_travel-city\_to, s\_travel-country\_to,

s\_travel-kms.  
ENDLOOP.

* Autres paramètres :

CONSTANTS: c\_year1 TYPE zcar\_brand\_mod-model\_year VALUE '2013',  
           c\_year2 TYPE zcar\_brand\_mod-model\_year VALUE '2016'.  
  
SELECT brand,  
       model  
  FROM zcar\_brand\_mod  
  INTO TABLE @DATA(t\_car)  
  WHERE model\_year BETWEEN @c\_year1 AND @c\_year2.  
  
  
DATA s\_car LIKE LINE OF t\_car.  
  
LOOP AT t\_car INTO s\_car.  
 WRITE:/ s\_car-brand, s\_car-model.  
ENDLOOP.

SELECT surname,  
       name  
  FROM zpassenger  
  INTO TABLE @DATA(t\_passenger)  
  WHERE ( surname LIKE '%EZ' OR surname LIKE '%ez' ).  
  
  
DATA s\_passenger LIKE LINE OF t\_passenger.  
  
LOOP AT t\_passenger INTO s\_passenger.  
 WRITE:/ s\_passenger-surname, s\_passenger-name.  
ENDLOOP.

SELECT surname,  
       name  
  FROM zpassenger  
  INTO TABLE @DATA(t\_passenger)  
  WHERE city NOT IN ('TOULOUSE', 'BARCELONE').  
  
  
DATA s\_passenger LIKE LINE OF t\_passenger.  
  
LOOP AT t\_passenger INTO s\_passenger.  
 WRITE:/ s\_passenger-surname, s\_passenger-name.  
ENDLOOP.

DATA: r\_year TYPE RANGE OF vlc\_year\_of\_construction.  
DATA: s\_year LIKE LINE OF r\_year.  
  
s\_year-sign   = 'I'.  
s\_year-option = 'BT'.  
s\_year-low    = '2013'.  
s\_year-high   = '2016'.  
APPEND s\_year TO r\_year.  
  
CLEAR s\_year.  
s\_year-sign   = 'I'.  
s\_year-option = 'EQ'.  
s\_year-low    = '2011'.  
APPEND s\_year TO r\_year.  
  
SELECT brand,  
       model  
  FROM zcar\_brand\_mod  
  INTO TABLE @DATA(t\_car)  
  WHERE model\_year IN @r\_year.  
  
  
DATA s\_car LIKE LINE OF t\_car.  
  
LOOP AT t\_car INTO s\_car.  
 WRITE:/ s\_car-brand, s\_car-model.  
ENDLOOP.

SELECT surname,  
       name,  
       date\_birth,  
       city,  
       country  
  FROM zdriver\_car AS d  
  INTO TABLE @DATA(t\_driver)  
  WHERE EXISTS ( SELECT \* FROM zpassenger  
                   WHERE surname = d~surname  
                     AND name    = d~name ).  
  
  
DATA s\_driver LIKE LINE OF t\_driver.  
  
LOOP AT t\_driver INTO s\_driver.  
  WRITE:/ s\_driver-surname, s\_driver-name, s\_driver-date\_birth,  
          s\_driver-city,    s\_driver-country.  
ENDLOOP.

SELECT city\_to && ', ' && country\_to AS city\_ctry  
  FROM ztravel AS t  
  INTO TABLE @DATA(t\_city)  
  WHERE city\_to NOT IN ( SELECT city  
                           FROM zdriver\_car  
                           WHERE city = t~city\_to )  
    AND city\_to NOT IN ( SELECT city  
                           FROM zpassenger  
                           WHERE city = t~city\_to ).  
  
  
DATA s\_city LIKE LINE OF t\_city.  
  
LOOP AT t\_city INTO s\_city.  
  WRITE:/ s\_city-city\_ctry.  
ENDLOOP.

* FOR ALL ENTRIES IN

SELECT brand,  
       model,  
       model\_year  
  FROM zcar\_brand\_mod  
  INTO TABLE @DATA(t\_car).  
  
IF NOT t\_car[] IS INITIAL.  
  
  SELECT car\_brand AS brand,  
         car\_model AS model,  
         car\_year  AS year,  
         surname,  
         name  
    FROM zdriver\_car  
    INTO TABLE @DATA(t\_driver)  
    FOR ALL ENTRIES IN @t\_car  
    WHERE car\_brand = @t\_car-brand  
      AND car\_model = @t\_car-model  
      AND car\_year  = @t\_car-model\_year.  
  
ENDIF.  
  
  
DATA s\_driver LIKE LINE OF t\_driver.  
  
LOOP AT t\_driver INTO s\_driver.  
 WRITE:/ s\_driver-brand, s\_driver-model, s\_driver-year,

S\_driver-surname, s\_driver-name.  
ENDLOOP.

* 1. AUTRES OPTIONS
* GROUP BY

SELECT d~surname               AS surname,  
       d~name                  AS name,  
       COUNT(\*)                AS nb\_travel,  
       SUM( t~kms )            AS kms,  
       SUM( t~gasol + t~toll ) AS costs  
  FROM ztravel AS t INNER JOIN zdriver\_car AS d  
  ON t~id\_driver = d~id\_driver  
  INTO TABLE @DATA(t\_travel)  
  GROUP BY d~surname, d~name.  
  
  
DATA s\_travel LIKE LINE OF t\_travel.  
  
LOOP AT t\_travel INTO s\_travel.  
  WRITE:/ s\_travel-surname, s\_travel-name, s\_travel-nb\_travel,  
          s\_travel-kms,     s\_travel-costs.  
ENDLOOP.

SELECT d~surname && @space && d~name     AS name,  
       COUNT(\*)                AS nb\_travel,  
       SUM( t~kms )            AS kms,  
       SUM( t~gasol + t~toll ) AS costs  
  FROM ztravel AS t INNER JOIN zdriver\_car AS d  
  ON t~id\_driver = d~id\_driver  
  INTO TABLE @DATA(t\_travel)  
  GROUP BY d~surname && @space && d~name.  
  
  
DATA s\_travel LIKE LINE OF t\_travel.  
  
LOOP AT t\_travel INTO s\_travel.  
  WRITE:/ s\_travel-name, s\_travel-nb\_travel,  
          s\_travel-kms,  s\_travel-costs.  
ENDLOOP.

* ORDER BY

SELECT d~surname               AS surname,  
       d~name                  AS name,  
       COUNT(\*)                AS nb\_travel,  
       SUM( t~kms )            AS kms,  
       SUM( t~gasol + t~toll ) AS costs  
  FROM ztravel AS t INNER JOIN zdriver\_car AS d  
  ON t~id\_driver = d~id\_driver  
  INTO TABLE @DATA(t\_travel)  
  GROUP BY d~surname, d~name

  ORDER BY d~surname ASCENDING, d~name ASCENDING.  
  
  
DATA s\_travel LIKE LINE OF t\_travel.  
  
LOOP AT t\_travel INTO s\_travel.  
  WRITE:/ s\_travel-surname, s\_travel-name, s\_travel-nb\_travel,  
          s\_travel-kms,     s\_travel-costs.  
ENDLOOP.

* HAVING

CONSTANTS: c\_nbretraj(2) TYPE i VALUE '2'.  
  
SELECT d~surname               AS surname,  
       d~name                  AS name,  
       COUNT(\*)                AS nb\_travel,  
       SUM( t~kms )            AS kms,  
       SUM( t~gasol + t~toll ) AS costs  
  FROM ztravel AS t INNER JOIN zdriver\_car AS d  
  ON t~id\_driver = d~id\_driver  
  INTO TABLE @DATA(t\_travel)  
  GROUP BY d~surname, d~name  
  HAVING COUNT(\*) GE @c\_nbretraj

  ORDER BY d~surname ASCENDING, d~name ASCENDING.  
  
  
DATA s\_travel LIKE LINE OF t\_travel.  
  
LOOP AT t\_travel INTO s\_travel.  
  WRITE:/ s\_travel-surname, s\_travel-name, s\_travel-nb\_travel,  
          s\_travel-kms,     s\_travel-costs.  
ENDLOOP.

* UP TO n ROWS

SELECT brand,  
       model  
  FROM zcar\_brand\_mod  
  UP TO 3 ROWS  
  INTO TABLE @DATA(t\_car)  
  ORDER BY brand DESCENDING.  
  
  
DATA s\_car LIKE LINE OF t\_car.  
  
LOOP AT t\_car INTO s\_car.  
 WRITE:/ s\_car-brand, s\_car-model.  
ENDLOOP.

1. INSERT

* Via une structure

DATA: s\_passager TYPE zpassenger.  
  
s\_passager-id\_passenger = 'P0005'.  
s\_passager-surname      = 'THIERRY'.  
s\_passager-name         = 'ROMAIN'.  
s\_passager-date\_birth   = '19930324'.  
s\_passager-city         = 'MONTPELLIER'.  
s\_passager-country      = 'FR'.  
s\_passager-lang         = 'F'.  
  
INSERT INTO zpassenger VALUES s\_passager.  
  
IF sy-subrc = 0.  
  WRITE 'Nouvel enregistrement créé avec succès'.  
ELSE.  
  WRITE 'Echec lors de la création d''un nouvel enregistrement'.  
ENDIF.

1. UPDATE

* SET

CONSTANTS: c\_id\_driver TYPE zdriver\_id VALUE 'C0003'.  
  
UPDATE ztravel  
      SET toll  = toll  + 5,  
          gasol = gasol + 10  
      WHERE id\_driver = @c\_id\_driver.  
  
  IF sy-subrc = 0.  
    WRITE 'Mise à jour réussie'.  
  ELSE.  
    WRITE 'Echec de la mise á jour'.  
  ENDIF.

* FROM structure

CONSTANTS: c\_id\_driver TYPE zdriver\_id VALUE 'C0001'.  
  
SELECT SINGLE \*  
  FROM ztravel  
  INTO @DATA(s\_travel)  
  WHERE id\_driver = @c\_id\_driver.  
  
IF sy-subrc = 0.  
  
  s\_travel-toll  = s\_travel-toll  + 5.  
  s\_travel-gasol = s\_travel-gasol + 10.  
  
  UPDATE ztravel FROM s\_travel.  
  
  IF sy-subrc = 0.  
    WRITE 'Mide à jour réussie'.  
  ELSE.  
    WRITE 'Echec de la mise á jour'.  
  ENDIF.

ENDIF.

* FROM TABLE

DATA: t\_travel TYPE TABLE OF ztravel.  
  
FIELD-SYMBOLS: <fs\_travel> TYPE ztravel.  
  
CONSTANTS: c\_id\_driver TYPE zdriver\_id VALUE 'C0003'.  
  
  
SELECT \*  
  FROM ztravel  
  INTO TABLE @t\_travel  
  WHERE id\_driver = @c\_id\_driver.  
  
IF sy-subrc = 0.  
  
  LOOP AT t\_travel ASSIGNING <fs\_travel>.  
  
    <fs\_travel>-toll  = <fs\_travel>-toll  - 5.  
    <fs\_travel>-gasol = <fs\_travel>-gasol - 10.  
  
  ENDLOOP.  
  
  UPDATE ztravel FROM TABLE t\_travel.  
  
  IF sy-subrc = 0.  
    WRITE 'Mise à jour réussie de la table ZTRAVEL'.  
  ELSE.  
    WRITE 'Echec de la mise á jour'.  
  ENDIF.  
  
ENDIF.

1. DELETE

CONSTANTS: c\_user\_id TYPE zpassenger\_id VALUE 'P0005'.  
  
DELETE FROM zpassenger WHERE id\_passenger = @c\_user\_id.  
  
IF sy-subrc = 0.  
  WRITE 'L''enregistrement a été supprimé avec succès'.  
ELSE.  
  WRITE 'Erreur lors de la suppression d''un enregistrement'.  
ENDIF.

1. INDEX

CONSTANTS: c\_city\_from TYPE ztravel-city\_from VALUE 'TOULOUSE',  
           c\_ctry\_from TYPE ztravel-country\_from VALUE 'FR'.  
  
SELECT city\_from,  
       country\_from,  
       city\_to,  
       country\_to,  
       date\_travel,  
       hour\_travel  
  FROM ztravel  
  INTO TABLE @DATA(t\_travel)  
  WHERE city\_from    = @c\_city\_from  
    AND country\_from = @c\_ctry\_from.