```
/* PROCEDURAL SQL IN SAS: */


/* PART-1:---> */


/* SQL STANDS FOR STRUCTURE QUERY LANGUAGE. */
/* IT IS USED TO QUERY THE DATABASES. */
/* IT IS ALSO KNOWN AS THE LANGUAGE OF DATABASES. */


/* PROC SQL: */
/* IT IS A POWERFUL PROCEDURE THAT COMBINES THE FUNCTIONALITY OF DATA AND PROC STEPS INTO A SINGLE STEP. */
/* PROC SQL CAN SORT, SUMMARIZE, SUBSET, JOIN(MERGE), AND CONCATENATE DATASETS, CREATE NEW VARIABLES,
AND PRINT THE RESULTS OR CREATE A NEW TABLE OR VIEW ALL IN ONE STEPS! */


/* SYNTAX:--> */

/* IT STARTS WITH

PROC SQL;

STATEMENT/STATEMENTS

/* QUIT; */  /* END WITH */



/* ELEMENTS OF PROC SQL: */
    /* SELECT  */
    /* FROM */
    /* WHERE */
    /* GROUP BY */
    /* HAVING */
    /* ORDER BY */


/* IT WILL GIVE THE MODEL,TYPES VARIABLES FROM THE CARS TABLE. */
```

```
PROC SQL;
SELECT MODEL, TYPE
FROM SASHELP.CARS;
QUIT;
```

```
/* PART-2:---> */

/* HOW TO SELECT ALL THE VARIABLES FROM A TABLE? */
```

```
PROC SQL;
SELECT *
FROM SASHELP.CARS;
QUIT;
```

```
/* HOW TO SELECT ONLY ONE VARIABLE FROM A TABLE? */
```

```
PROC SQL;
SELECT MODEL
FROM SASHELP.CARS;
QUIT;
```

```
/* HOW TO SELECT MORE THAN ONE VARIABLE FROM A TABLE? */
```

```
PROC SQL;
SELECT MODEL, TYPE, ORIGIN, MSRP
FROM SASHELP.CARS;
QUIT;
```

```
/* HOW TO FILTER THE DATA?   */

PROC SQL;
SELECT MODEL, TYPE, ORIGIN, MSRP
FROM SASHELP.CARS
WHERE TYPE= "Sedan" AND ORIGIN="Asia" ;
QUIT;



/* HOW TO SUMMARIZE THE DATA? */

/* EG-1: */

/* SUMMARIZE THE DATA BY TYPE,ORIGIN */

PROC SQL;
SELECT  TYPE,
        ORIGIN,

        SUM(MSRP) AS TOTAL_MSRP
        FROM SASHELP.CARS
        GROUP BY TYPE,ORIGIN;

QUIT;

/* EG-2: */

/* SUMMARIZE THE DATA BY TYPE */

PROC SQL;
SELECT  TYPE,
        COUNT(TYPE) AS TOTAL_UNITS,
        SUM(MSRP) AS TOTAL_MSRP
        FROM SASHELP.CARS
        GROUP BY TYPE;

QUIT;


/* EG-3: */

/* SUMMARIZE THE DATA BY TYPE WITH WHERE CONDITION */

PROC SQL;
SELECT  TYPE,
        COUNT(TYPE) AS TOTAL_UNITS,
        SUM(MSRP) AS TOTAL_MSRP
        FROM SASHELP.CARS
        WHERE MSRP>=60000
        GROUP BY TYPE;

QUIT;


/* EG-4: */

/* SUMMARIZE THE DATA BY TYPE WITH HAVING CLAUSE */

PROC SQL;
SELECT  TYPE,
        COUNT(TYPE) AS TOTAL_UNITS,
        SUM(MSRP) AS TOTAL_MSRP
        FROM SASHELP.CARS
        WHERE MSRP>=60000
        GROUP BY TYPE
        HAVING TOTAL_UNITS<=50;

QUIT;


/* HOW TO USE ORDER BY CLAUSE? */
```

```
PROC SQL;
SELECT  TYPE,
        COUNT(TYPE) AS TOTAL_UNITS,
        SUM(MSRP) AS TOTAL_MSRP
        FROM SASHELP.CARS
        GROUP BY TYPE
        ORDER BY TYPE DESC;

QUIT;



/* PART-3:--> */


/* CASE WHEN: */
/* IT HELPS IN CONDITIONALLY SELECTING RESULT VALUE FROM EACH ROW/OBSERVATION IN A TABLE. */
/* IT HELPS IN CREATING VARIABLES BASED ON CERTAIN CONDITIONS. */
```

```
PROC SQL;
SELECT
        CUSTOMER_ID,
        Company,
        GENDER,
        Age,
        CASE
            WHEN AGE>=60 THEN 'OLD_AGE'
            WHEN AGE>=40 THEN 'MID_AGE'
            WHEN AGE>=30 THEN 'MID_YOUNG_AGE'
            ELSE 'YOUNG_AGE'
        END AS AGE_BUCKET
        FROM WORK.SCAS;
QUIT;
```

```


/* PART-4:--> */


/* JOINS IN PROC SQL: */

/* TYPES OF JOINS: */

/* a. LEFT JOIN */
/* b. RIGHT JOIN */
/* c. INNER JOIN */
/* d. FULL JOIN */


/* a. LEFT JOIN: */
```

```
PROC SQL;
        SELECT
        A.NAME,
        A.SEX,
        A.AGE,
        B.HEIGHT,
        B.WEIGHT
        FROM WORK.PERS_INFO AS A
        LEFT JOIN
        WORK.PHY_INFO AS B
        ON A.NAME = B.STD_NAME;

QUIT;


/* b. RIGHT JOIN: */
```

```
PROC SQL;
        SELECT
        A.NAME,
```

```
        A.SEX,
        A.AGE,
        B.HEIGHT,
        B.WEIGHT
        FROM WORK.PERS_INFO AS A
        RIGHT JOIN
        WORK.PHY_INFO AS B
        ON A.NAME = B.STD_NAME;

QUIT;


/* c. INNER JOIN: */

PROC SQL;
        SELECT
        A.NAME,
        A.SEX,
        A.AGE,
        B.HEIGHT,
        B.WEIGHT
        FROM WORK.PERS_INFO AS A
        INNER JOIN
        WORK.PHY_INFO AS B
        ON A.NAME = B.STD_NAME;

QUIT;


/* d. FULL JOIN: */

PROC SQL;
        SELECT
        A.NAME,
        A.SEX,
        A.AGE,
        B.HEIGHT,
        B.WEIGHT
        FROM WORK.PERS_INFO AS A
        FULL JOIN
        WORK.PHY_INFO AS B
        ON A.NAME = B.STD_NAME;

QUIT;




/* PART-5:---> */


/* CROSS JOIN: */
/* IT IS ALSO KNOWN AS CARTESIAN JOIN. */
/* IT IS THE SIMPLEST OF ALL THE POSSIBLE JOINS TO WRITE AND UNDERSTAND. */
/* IT'S SIMPLY A COMBINATION OF TWO TABLES. */
/* IT'S RESULT IS SIMPLY EACH ROW FROM FIRST TABLE AND EACH ROW FROM SECOND TABLE. */
/* EG:--> IF THE FIRST TABLE HAS 3 OBSERVATIONS AND SECOND TABLE HAS 2 OBSERVATIONS
            THEN THE RESULATING TABLE WOULD HAVE 6 OBSERVATIONS. */

/* EG:--> YOU HAVE GIVEN 2 VARIABLES 'X' AND 'Y'WHERE IN MULTIPLY WITH 2.
            (X+Y)*2=2X+2Y.
            SAME THING HAPPEND WITH CROSS JOIN */

PROC SQL;
SELECT
    A.*,
    B.HEIGHT
    FROM
    WORK.CART1 AS A
    CROSS JOIN
    WORK.CART2 AS B;

QUIT;
```

```
/* HOW TO CREATE A TABLE? */

PROC SQL;
CREATE TABLE CROS_COMBINE AS
SELECT
    A.*,
    B.HEIGHT
    FROM
    WORK.CART1 AS A
    CROSS JOIN
    WORK.CART2 AS B;
```