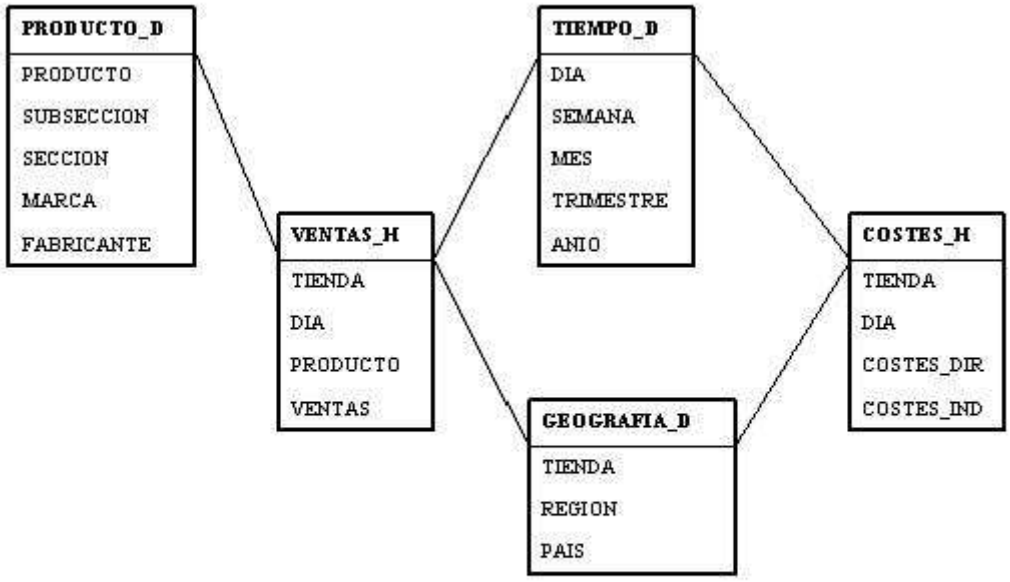
Es un ejemplo de una cadena de tiendas, por ejemplo LeroyMerlin, tenemos dos tablas de hechos, Ventas y Costes con sus tablas de dimensiones.



Ejercicio1: Modelar Ventas con dos dimensiones (Geografía y Producto)

Creamos un usuario para el ejercicio y creamos las tablas:

```
CREATE TABLE GEOGRAFIA_D
(TIENDA VARCHAR(20) PRIMARY KEY, REGION VARCHAR(20), PAIS VARCHAR(20));

CREATE TABLE VENTAS_H (TIENDA VARCHAR(20),
PRODUCTO VARCHAR(20), VENTAS NUMBER(20),
 PRIMARY KEY (TIENDA, PRODUCTO),
 CONSTRAINT FK1 FOREIGN KEY (TIENDA) REFERENCES GEOGRAFIA_D (TIENDA));

INSERT INTO GEOGRAFIA_D VALUES ('RIVAS', 'SUR', 'ESPAÑA');
INSERT INTO GEOGRAFIA_D VALUES ('ALCORCON', 'NORTE', 'FRANCIA');
INSERT INTO GEOGRAFIA_D VALUES ('LAS ROZAS', 'NORTE', 'ESPAÑA');
INSERT INTO GEOGRAFIA_D VALUES ('MAJADAHONDA', 'NORTE', 'ESPAÑA');
INSERT INTO VENTAS_H VALUES ('ALCORCON', 'MESAS JARDIN', 150);
INSERT INTO VENTAS_H VALUES ('ALCORCON', 'MUEBLES BAÑO', 60);
INSERT INTO VENTAS_H VALUES ('LAS ROZAS', 'MESAS JARDIN', 200);
INSERT INTO VENTAS_H VALUES ('LAS ROZAS', 'MUEBLES BAÑO', 50);
INSERT INTO VENTAS_H VALUES ('MAJADAHONDA', 'MESAS JARDIN', 50);
INSERT INTO VENTAS_H VALUES ('MAJADAHONDA', 'MUEBLES BAÑO', 90);
INSERT INTO VENTAS_H VALUES ('RIVAS', 'MESAS JARDIN', 90);
INSERT INTO VENTAS_H VALUES ('RIVAS', 'MUEBLES BAÑO', 150);
```

Quedaría de la siguiente forma:

TIENDA	PRODUCTO	VENTAS
ALCORCON	MESAS DE JARDIN	150
ALCORCON	MUEBLES DE BAÑO	60
LAS ROZAS	MESAS DE JARDIN	200
LAS ROZAS	MUEBLES DE BAÑO	50
MAJADAHONDA	MESAS DE JARDIN	50
MAJADAHONDA	MUEBLES DE BAÑO	90
RIVAS	MESAS DE JARDIN	90
RIVAS	MUEBLES DE BAÑO	150

TIENDA	REGION	PAIS
RIVAS	SUR	ESPAÑA
ALCORCON	NORTE	FRANCIA
MAJADAHONDA	NORTE	ESPAÑA
LAS ROZAS	NORTE	ESPAÑA

¿Cómo puedo generar el cubo de ventas? Con sus subtotales y totales:

	ALCORCON	LAS ROZAS	MAJADAHONDA	RIVAS	Total general
MESAS DE JARDIN	150	200	50	90	490
MUEBLES DE BAÑO	60	50	90	150	350
Total general	210	250	140	240	840

A la tabla VENTAS_H debemos añadirle los registros de totales, nos faltan 4x2=6 subtotales y el gran total, es decir 7 registros

```
SELECT TIENDA, PRODUCTO, VENTAS FROM VENTAS_H
UNION ALL /* SUBTOTALES POR PRODUCTO  2*/
SELECT NULL, PRODUCTO, SUM(VENTAS) FROM VENTAS_H
GROUP BY  PRODUCTO
UNION ALL /* SUBTOTALES POR TIENDA  4*/
SELECT TIENDA,NULL, SUM(VENTAS) FROM VENTAS_H
GROUP BY  TIENDA
UNION ALL /*TOTAL 1 */
SELECT NULL,NULL, SUM(VENTAS) FROM VENTAS_H;
```

TIENDA	PRODUCTO	VENTAS
ALCORCON	MESAS JARDIN	150
ALCORCON	MUEBLES BAÑO	60
ALCORCON		210
LAS ROZAS	MESAS JARDIN	200
LAS ROZAS	MUEBLES BAÑO	50
LAS ROZAS		250
MAJADHONDA	MESAS JARDIN	50
MAJADHONDA	MUEBLES BAÑO	90
MAJADHONDA		140
RIVAS	MESAS JARDIN	90
RIVAS	MUEBLES BAÑO	150
RIVAS		240
	MESAS JARDIN	490
	MUEBLES BAÑO	350
		840

15 filas seleccionadas.

Para evitar hacer esto, Oracle dispone de unas extensiones de SQL para agregaciones en DataWarehouses, por ejemplo **CUBE**:

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS) FROM VENTAS_H
GROUP BY
  CUBE (TIENDA, PRODUCTO);
```

TIENDA	PRODUCTO	SUM(VENTAS)
		840
	MESAS JARDIN	490
	MUEBLES BAÑO	350
RIVAS		240
RIVAS	MESAS JARDIN	90
RIVAS	MUEBLES BAÑO	150
ALCORCON		210
ALCORCON	MESAS JARDIN	150
ALCORCON	MUEBLES BAÑO	60
LAS ROZAS		250
LAS ROZAS	MESAS JARDIN	200
LAS ROZAS	MUEBLES BAÑO	50
MAJADHONDA		140
MAJADHONDA	MESAS JARDIN	50
MAJADHONDA	MUEBLES BAÑO	90

15 filas seleccionadas.

ROLLUP, se usa más para navegar en las jerarquías aunque también puede usarse con dimensiones, por ejemplo:

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS) FROM VENTAS_H
GROUP BY
  ROLLUP (TIENDA, PRODUCTO);
```

TIENDA	PRODUCTO	SUM(VENTAS)
RIVAS	MESAS JARDIN	90
RIVAS	MUEBLES BAÑO	150
RIVAS		240
ALCORCON	MESAS JARDIN	150
ALCORCON	MUEBLES BAÑO	60
ALCORCON		210
LAS ROZAS	MESAS JARDIN	200
LAS ROZAS	MUEBLES BAÑO	50
LAS ROZAS		250
MAJADHONDA	MESAS JARDIN	50
MAJADHONDA	MUEBLES BAÑO	90
MAJADHONDA		140
		840

13 filas seleccionadas.

Sin embargo faltan dos filas ¿cuáles? Los subtotales de producto. Nos da los agregados de tienda para cada una de las siguientes columnas, pero nunca nos dará los agregados del resto de dimensiones.

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS) FROM VENTAS_H
GROUP BY
  ROLLUP (PRODUCTO, TIENDA); /* INTERCAMBIAMOS EL ORDEN */
```

TIENDA	PRODUCTO	SUM(VENTAS)
RIVAS	MESAS JARDIN	90
ALCORCON	MESAS JARDIN	150
LAS ROZAS	MESAS JARDIN	200
MAJADHONDA	MESAS JARDIN	50
	MESAS JARDIN	490
RIVAS	MUEBLES BAÑO	150
ALCORCON	MUEBLES BAÑO	60
LAS ROZAS	MUEBLES BAÑO	50
MAJADHONDA	MUEBLES BAÑO	90
	MUEBLES BAÑO	350
		840

11 filas seleccionadas.

Nos falta los subtotales por tienda... No es un cubo completo, es muy útil para las jerarquías.

```
SELECT  PAIS , REGION  , H.TIENDA , SUM(VENTAS)  FROM VENTAS_H H, GEOGRAFIA_D D
WHERE H.TIENDA=D.TIENDA
GROUP BY
  ROLLUP (PAIS, REGION, H.TIENDA);
```

PAIS	REGION	TIENDA	SUM(VENTAS)
ESPAÑA	SUR	RIVAS	240
ESPAÑA	SUR		240
ESPAÑA	NORTE	LAS ROZAS	250
ESPAÑA	NORTE	MAJADAHONDA	140
ESPAÑA	NORTE		390
ESPAÑA			630
FRANCIA	NORTE	ALCORCON	210
FRANCIA	NORTE		210
FRANCIA			210
			840

10 filas seleccionadas.

Se puede hacer parcial (ver que falta el total por país), por ejemplo:

```
SELECT  PAIS , REGION  , H.TIENDA , SUM(VENTAS)  FROM VENTAS_H H, GEOGRAFIA_D D
WHERE H.TIENDA=D.TIENDA
GROUP BY PAIS,
  ROLLUP (REGION, H.TIENDA);
```

PAIS	REGION	TIENDA	SUM(VENTAS)
ESPAÑA	SUR	RIVAS	240
ESPAÑA	SUR		240
ESPAÑA	NORTE	LAS ROZAS	250
ESPAÑA	NORTE	MAJADAHONDA	140
ESPAÑA	NORTE		390
ESPAÑA			630
FRANCIA	NORTE	ALCORCON	210
FRANCIA	NORTE		210
FRANCIA			210

9 filas seleccionadas.

También el cubo se puede hacer parcial:

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS) FROM VENTAS_H
GROUP BY TIENDA,
  CUBE (PRODUCTO);
```

TIENDA	PRODUCTO	SUM(VENTAS)
RIVAS		240
RIVAS	MESAS JARDIN	90
RIVAS	MUEBLES BAÑO	150
ALCORCON		210
ALCORCON	MESAS JARDIN	150
ALCORCON	MUEBLES BAÑO	60
LAS ROZAS		250
LAS ROZAS	MESAS JARDIN	200
LAS ROZAS	MUEBLES BAÑO	50
MAJADAHONDA		140
MAJADAHONDA	MESAS JARDIN	50
MAJADAHONDA	MUEBLES BAÑO	90

12 filas seleccionadas.

Ahora surge el problema de determinar cuales de los registros son subtotales o totales, y cual es el nivel exacto de agregación. Si queremos calcular un % sobre el total, debemos ser capaces de encontrar el total con facilidad. Además qué ocurre si uno de los registros tiene valores nulos en su dimensión ¿cómo lo distinguimos de un subtotal?

Funciones GROUPING

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS),
GROUPING (TIENDA) G_TIENDA, GROUPING(PRODUCTO) G_PRODUCTO
FROM VENTAS_H
GROUP BY
  CUBE (TIENDA ,PRODUCTO);
```

TIENDA	PRODUCTO	SUM(VENTAS)	G_TIENDA	G_PRODUCTO
		840	1	1
	MESAS JARDIN	490	1	0
	MUEBLES BAÑO	350	1	0
RIVAS		240	0	1
RIVAS	MESAS JARDIN	90	0	0
RIVAS	MUEBLES BAÑO	150	0	0
ALCORCON		210	0	1
ALCORCON	MESAS JARDIN	150	0	0
ALCORCON	MUEBLES BAÑO	60	0	0
LAS ROZAS		250	0	1
LAS ROZAS	MESAS JARDIN	200	0	0
LAS ROZAS	MUEBLES BAÑO	50	0	0
MAJADAHONDA		140	0	1
MAJADAHONDA	MESAS JARDIN	50	0	0
MAJADAHONDA	MUEBLES BAÑO	90	0	0

15 filas seleccionadas.

Podemos ver que los registros que tienen 1 significa que está totalizada esa dimensión y los que tienen 0 no. Así, el registro con 1 1 significará el gran total, y un 1 0 un subtotal, y las 0 0 serán las celdas del cubo.

Así podríamos escribir una select que basándose en estos valores completara los nulos:

```
SELECT DECODE(GROUPING (TIENDA) , 1, 'AGR Tienda', TIENDA), DECODE(GROUPING
(PRODUCTO) , 1, 'AGR Producto', PRODUCTO), SUM(VENTAS)
FROM VENTAS_H
GROUP BY
  CUBE (TIENDA ,PRODUCTO);
```

DECODE(GROUPING(TIENDA),1,'AG	DECODE(GROUPING(PRODUCTO),1,'	SUM(VENTAS)
AGR Tienda	AGR Producto	840
AGR Tienda	MESAS JARDIN	490
AGR Tienda	MUEBLES BAÑO	350
RIVAS	AGR Producto	240
RIVAS	MESAS JARDIN	90
RIVAS	MUEBLES BAÑO	150
ALCORCON	AGR Producto	210
ALCORCON	MESAS JARDIN	150
ALCORCON	MUEBLES BAÑO	60
LAS ROZAS	AGR Producto	250
LAS ROZAS	MESAS JARDIN	200
LAS ROZAS	MUEBLES BAÑO	50
MAJADAHONDA	AGR Producto	140
MAJADAHONDA	MESAS JARDIN	50
MAJADAHONDA	MUEBLES BAÑO	90

15 filas seleccionadas.

También es muy útil al combinarla con la cláusula HAVING para obtener solo un tipo de resultado. Por ejemplo totales y subtotales:

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS),
GROUPING (TIENDA) G_TIENDA, GROUPING(PRODUCTO) G_PRODUCTO
FROM VENTAS_H
GROUP BY
  CUBE (TIENDA ,PRODUCTO)
HAVING GROUPING (TIENDA)=1 OR GROUPING(PRODUCTO)=1;
```

TIENDA	PRODUCTO	SUM(VENTAS)	G_TIENDA	G_PRODUCTO
	MESAS JARDIN	490	1	0
	MUEBLES BAÑO	350	1	0
ALCORCON		210	0	1
LAS ROZAS		250	0	1
MAJADAHONDA		140	0	1
RIVAS		240	0	1
		840	1	1

7 filas seleccionadas.

Con esta misma filosofía está el GROUPING_ID y nace con el objeto de reducir el espacio de almacenamiento (1 columna por dimensión frente a una sola por conjunto de dimensiones). Se calcula construyendo un número en binario siguiendo el orden de las dimensiones marcadas.

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS),
GROUPING (TIENDA) G_TIENDA, GROUPING(PRODUCTO) G_PRODUCTO,
GROUPING_ID (TIENDA ,PRODUCTO) GI
FROM VENTAS_H
GROUP BY CUBE (TIENDA ,PRODUCTO);
```

TIENDA	PRODUCTO	SUM(VENTAS)	G_TIENDA	G_PRODUCTO	GI
		840	1	1	3
	MESAS JARDIN	490	1	0	2
	MUEBLES BAÑO	350	1	0	2
RIVAS		240	0	1	1
RIVAS	MESAS JARDIN	90	0	0	0
RIVAS	MUEBLES BAÑO	150	0	0	0
ALCORCON		210	0	1	1
ALCORCON	MESAS JARDIN	150	0	0	0
ALCORCON	MUEBLES BAÑO	60	0	0	0
LAS ROZAS		250	0	1	1
LAS ROZAS	MESAS JARDIN	200	0	0	0
LAS ROZAS	MUEBLES BAÑO	50	0	0	0
MAJADAHONDA		140	0	1	1
MAJADAHONDA	MESAS JARDIN	50	0	0	0
MAJADAHONDA	MUEBLES BAÑO	90	0	0	0

15 filas seleccionadas.

/* 17 DE FEBRERO */

GROUPING SETS

Permiten decidir exactamente las dimensiones que entran en juego sin computar el cubo entero.

CUBE(a, b, c)
=
GROUPING SETS ((a, b, c), (a, b), (a, c), (b, c), (a), (b), (c), ())

ROLLUP(a, b, c)
=
GROUPING SETS ((a, b, c), (a, b), ())

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS)
FROM VENTAS_H
GROUP BY
  GROUPING SETS ( (TIENDA,PRODUCTO), (TIENDA));
```

Mostrará todas las combinaciones TiendaXProducto 8, y los totales por tienda 4, es decir 12 registros.

TIENDA	PRODUCTO	SUM(VENTAS)
RIVAS	MESAS JARDIN	90
RIVAS	MUEBLES BAÑO	150
RIVAS		240
ALCORCON	MESAS JARDIN	150
ALCORCON	MUEBLES BAÑO	60
ALCORCON		210
LAS ROZAS	MESAS JARDIN	200
LAS ROZAS	MUEBLES BAÑO	50
LAS ROZAS		250
MAJADAHONDA	MESAS JARDIN	50
MAJADAHONDA	MUEBLES BAÑO	90
MAJADAHONDA		140

12 filas seleccionadas.

Con lo que hemos visto hasta el momento podemos:

- Generar cubos
- Operar con los cubos:
  - Exploración ascendente (roll-up) ¿Podemos subir un nivel en una jerarquía? Un simple join y agregación. Tenemos las ventas por tienda y las queremos por región??
  - Exploración descendente (drill-down) Desciende un nivel en una jerarquía de dimensión. Tenemos las ventas por región y queremos las ventas por tienda.
  - Proyectar (slice&dice). Reduce dimensiones en un tabla, fija un valor de una o varias dimensiones. Queremos solo los de la tienda de Rivas, un simple filtro.
  - Pivotación (pivot). Se rota la presentación del esquema de hecho (sin cambiarlo). Cambio la definición del cubo
- Funciones de agregación: Además de las conocidas, estadísticas y de agregación, Oracle dispone de *"analytic SQL functions"* que permiten

Table 21-1 Analytic Functions and Their Uses

Type	Used For
Ranking	Calculating ranks, percentiles, and n-tiles of the values in a result set.
Windowing	Calculating cumulative and moving aggregates. Works with these functions: `SUM`, `AVG`, `MIN`, `MAX`, `COUNT`, `VARIANCE`, `STDDEV`, `FIRST_VALUE`, `LAST_VALUE`, and new statistical functions. Note that the `DISTINCT` keyword is not supported in windowing functions except for `MAX` and `MIN`.
Reporting	Calculating shares, for example, market share. Works with these functions: `SUM`, `AVG`, `MIN`, `MAX`, `COUNT` (with/without `DISTINCT`), `VARIANCE`, `STDDEV`, `RATIO_TO_REPORT`, and new statistical functions. Note that the `DISTINCT` keyword may be used in those reporting functions that support `DISTINCT` in aggregate mode.
`LAG/LEAD`	Finding a value in a row a specified number of rows from a current row.
`FIRST/LAST`	First or last value in an ordered group.
Linear Regression	Calculating linear regression and other statistics (slope, intercept, and so on).
Inverse Percentile	The value in a data set that corresponds to a specified percentile.
Hypothetical Rank and Distribution	The rank or percentile that a row would have if inserted into a specified data set.

Veamos unas de las más usadas Ranking, tenemos dos funciones RANK y DENSE_RANK, son muy parecidas, la diferencia es, si tenemos los siguientes datos: 100,100,50,20
Dense Rank diría que en la primera posición están 100 y 100, en la segunda 50 y en la cuarta 20
Rank diría que en la primera posición están 100 y 100, en la tercera 50 y en la cuarta 20

Por ejemplo:

```
SELECT TIENDA, PRODUCTO, SUM(VENTAS), DENSE_RANK() OVER (ORDER BY SUM(VENTAS) DESC) D_RANK, RANK() OVER (ORDER BY SUM(VENTAS) DESC) RANK
FROM VENTAS_H
GROUP BY TIENDA, PRODUCTO;
```

TIENDA	PRODUCTO	SUM(VENTAS)	D_RANK	RANK
LAS ROZAS	MESAS JARDIN	200	1	1
RIVAS	MUEBLES BAÑO	150	2	2
ALCORCON	MESAS JARDIN	150	2	2
RIVAS	MESAS JARDIN	90	3	4
MAJADAHONDA	MUEBLES BAÑO	90	3	4
ALCORCON	MUEBLES BAÑO	60	4	6
LAS ROZAS	MUEBLES BAÑO	50	5	7
MAJADAHONDA	MESAS JARDIN	50	5	7

8 filas seleccionadas.

Ejercicio2: Completar el modelo para VENTAS_H. Introducir la dimensión tiempo y producto en sus dos vertientes (es una dimensión que tiene dos jerarquías). Insertar valores en las nuevas dimensiones y en ventas.

Practicar con CUBE, ROLLUP, GROUPING SETS y utilizar las funciones GROUPING, GROUPING ID y rankings.

Creamos el modelo de datos y lo rellenamos de datos:

```
CREATE TABLE PRODUCTO_D
(PRODUCTO VARCHAR(20) PRIMARY KEY,
SUBSECCION VARCHAR(20),
SECCION VARCHAR(20),
MARCA VARCHAR(20),
FABRICANTE VARCHAR(20));

INSERT INTO PRODUCTO_D VALUES ('MESAS JARDIN','MOBILIARIO JARDIN','JARDIN','TODO
JARDIN','MUEBLES PACO');
INSERT INTO PRODUCTO_D VALUES ('MUEBLES BAÑO','EQUIPAMIENTO
BAÑOS','SANITARIO','BELLAVISTA','ROCA');

ALTER TABLE VENTAS_H ADD
CONSTRAINT FK2 FOREIGN KEY (PRODUCTO) REFERENCES PRODUCTO_D (PRODUCTO);

CREATE TABLE TIEMPO_D
( DIA VARCHAR(8)  PRIMARY KEY,
FECHA DATE,
DIA_SEMANA VARCHAR(10),
SEMANA NUMBER(1),
MES NUMBER(2),
N_MES VARCHAR(10),
TRIMESTRE NUMBER(1),
TRIM VARCHAR(2),
ANYO NUMBER(4));

DECLARE
 f DATE;
BEGIN
FOR i IN 1..500 LOOP
  f:=SYSDATE-i;
  INSERT INTO TIEMPO_D VALUES
(
   TO_CHAR(f,'YYYY')||TO_CHAR(f,'MM')||TO_CHAR(f,'DD'),
   TRUNC(f),
   TO_CHAR(f,'DAY'),
   TO_NUMBER(TO_CHAR(f,'W')), /*semana del mes*/
   TO_NUMBER(TO_CHAR(f,'MM')),
   TO_CHAR(f,'MONTH'),
   TO_NUMBER(TO_CHAR(f,'Q')),
   'Q'||TO_CHAR(f,'Q'),
   TO_NUMBER(TO_CHAR(f,'YYYY'))
);
```

```
END LOOP;
END;

ALTER TABLE VENTAS_H ADD DIA VARCHAR(8);

UPDATE VENTAS_H SET DIA=TO_CHAR(sysdate-1,'YYYY')||TO_CHAR(sysdate-
1,'MM')||TO_CHAR(sysdate-1,'DD');

ALTER TABLE VENTAS_H ADD
CONSTRAINT FK3 FOREIGN KEY (DIA) REFERENCES TIEMPO_D (DIA);

ALTER TABLE VENTAS_H MODIFY PRIMARY KEY (TIENDA, PRODUCTO, DIA);

/* si esta sentencia da problemas se puede borrar la PK de VENTAS_H y volverla a crear, si
desconocemos el nombre de la PK basta con buscarla en la vista user_constraints */

DECLARE
f DATE;
c VARCHAR(8);
BEGIN
FOR i IN 2..500 LOOP
  f:=SYSDATE-i;
  c:=TO_CHAR(f,'YYYY')||TO_CHAR(f,'MM')||TO_CHAR(f,'DD');
  INSERT INTO VENTAS_H VALUES ('ALCORCON', 'MESAS
JARDIN',dbms_random.value(0,200),c);
  INSERT INTO VENTAS_H VALUES ('ALCORCON', 'MUEBLES BAÑO',
dbms_random.value(0,200),c);
  INSERT INTO VENTAS_H VALUES ('LAS ROZAS', 'MESAS JARDIN',
dbms_random.value(0,200),c);
  INSERT INTO VENTAS_H VALUES ('LAS ROZAS', 'MUEBLES BAÑO',
dbms_random.value(0,200),c);
  INSERT INTO VENTAS_H VALUES ('MAJADAHONDA', 'MESAS JARDIN',
dbms_random.value(0,200),c);
  INSERT INTO VENTAS_H VALUES ('MAJADAHONDA', 'MUEBLES BAÑO',
dbms_random.value(0,200),c);
  INSERT INTO VENTAS_H VALUES ('RIVAS', 'MESAS JARDIN', dbms_random.value(0,200),c);
  INSERT INTO VENTAS_H VALUES ('RIVAS', 'MUEBLES BAÑO', dbms_random.value(0,200),c);
END LOOP;
END;

select count(*) from ventas_h; /* 4000 registros */
```

A partir de aquí los alumnos pueden practicar del mismo modo que en el ejercicio anterior pero con un modelo más complejo y con más datos.