COMP40725 Introduction to Relational Databases and SQL Programming

Final Project

14205954_Jin Yao

Project Idea
I wanna design a database that is going to be used to store a company's data. The database is required to store data about the company's products, as well as customer, staff, wholesaler and order details.

Business Rules

There are various business rules that need be accounted for within the database:

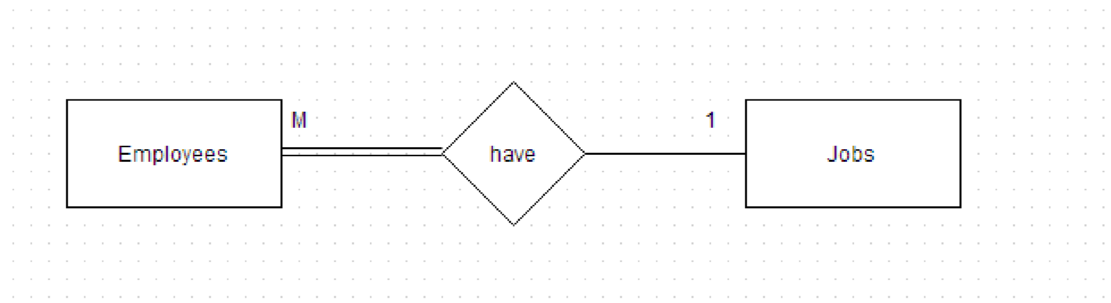
- Multiple employees can be assigned to a manager
- Each employee must and only have one manager
- Each customer can be assigned to multiple staff members
- An employee can be assigned to multiple customers
- A customer can only be assigned to one employee at a time
- It is optional for a customer to be assigned to an employee
- It is optional for a employee to have an customer
- Each product must and only have one category
- Many products can have the same category
- A category can exist without being associated with any product
- A product must and only come from a wholesaler
- Many products can come from the same wholesaler
- A wholesaler can exist without being associated with any product
- An order must include at least one product.
- The same product can be ordered numerous times on one individual order
- A product can exist without being on any order
- Many orders can contain the same products
- Each order must and can only involve one customer
- A customer can place multiple orders
- A customer can exist without placing any orders
- Each employee must and only be assigned to one job
- Multiple employees can do the same job
- A job can exist without having any employee in that role
- One manager can be in charge of multiple employees
- An employee must have a manager
- An employee may be his/her own manager
- An employee can only have one manager

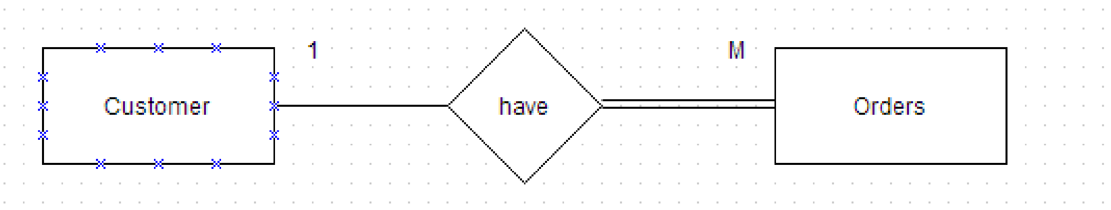
- A department must be managed by an employee
- An employee can manage a department without being a manager
- A department can be managed by multiple employees
- An employee can be in charge of multiple departments
- An employee can exist without being in charge of a department

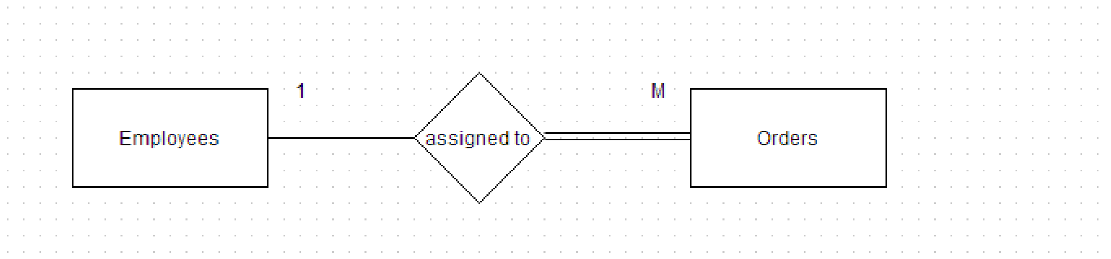
Identify entites, their attributes and primary keys

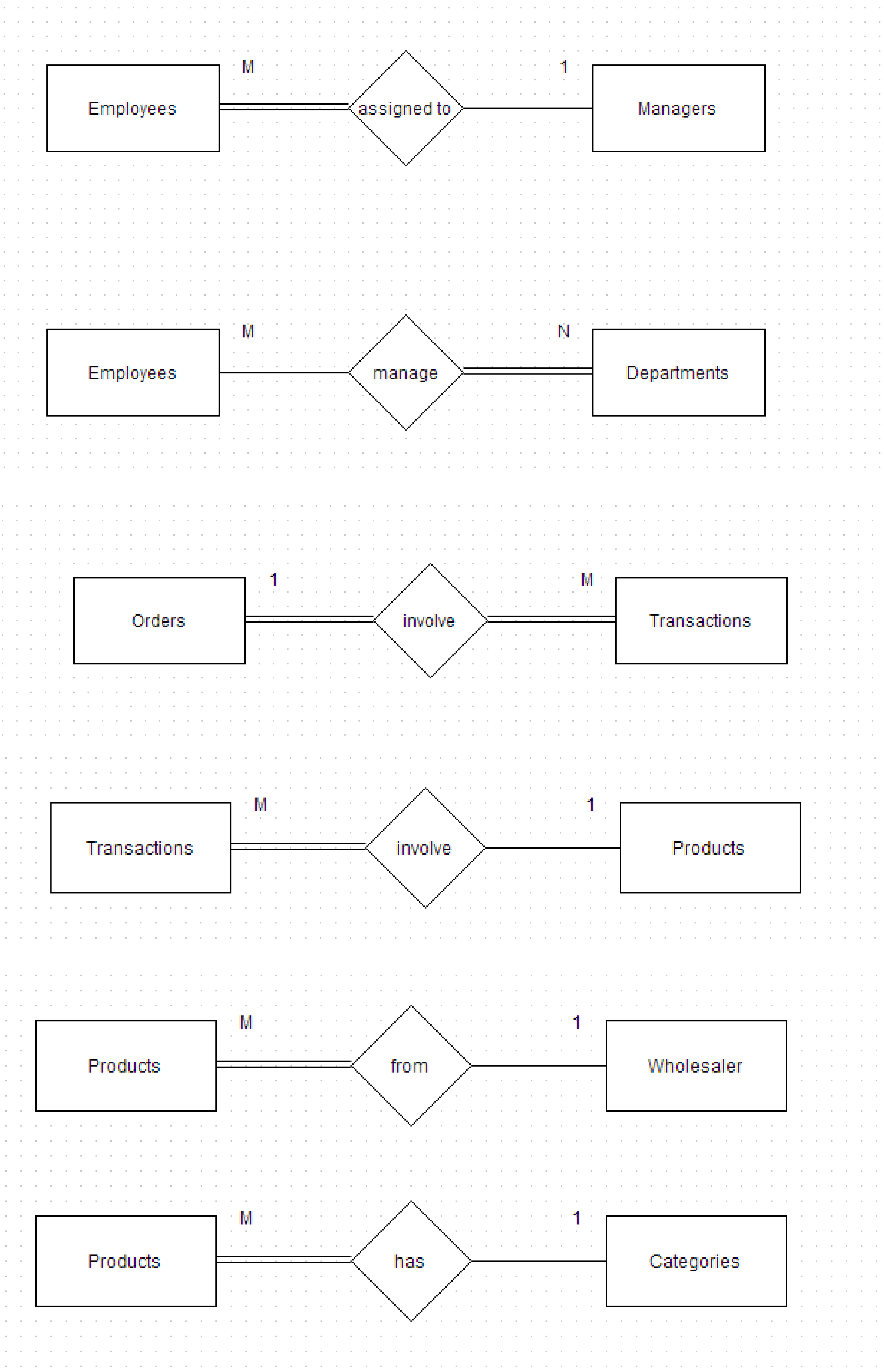
Customers (cust_id, cust_fname, cust_lname, cust_tel)
Employees (emp_id, emp_fname, emp_lname, emp_tel, job_id, manager_emp_id)
Departments (dept_id, dept_name, manager_emp_id)
Jobs (job_id, job_title, job_salary)
Managers (manager_emp_id, manager_fname, manager_ lname)
Orders (order_id, order_date, cust_id, emp_id)
Transactions (order_id, prod_id, prod_quantity)
Products (prod_id, prod_name, cat_id, wholesaler_id, prod_cost, prod_sale_price)
Categories (cat_id, cat_name)
Wholesalers(wholesaler_id, wholesaler_name)

Identify relationships and cardinality

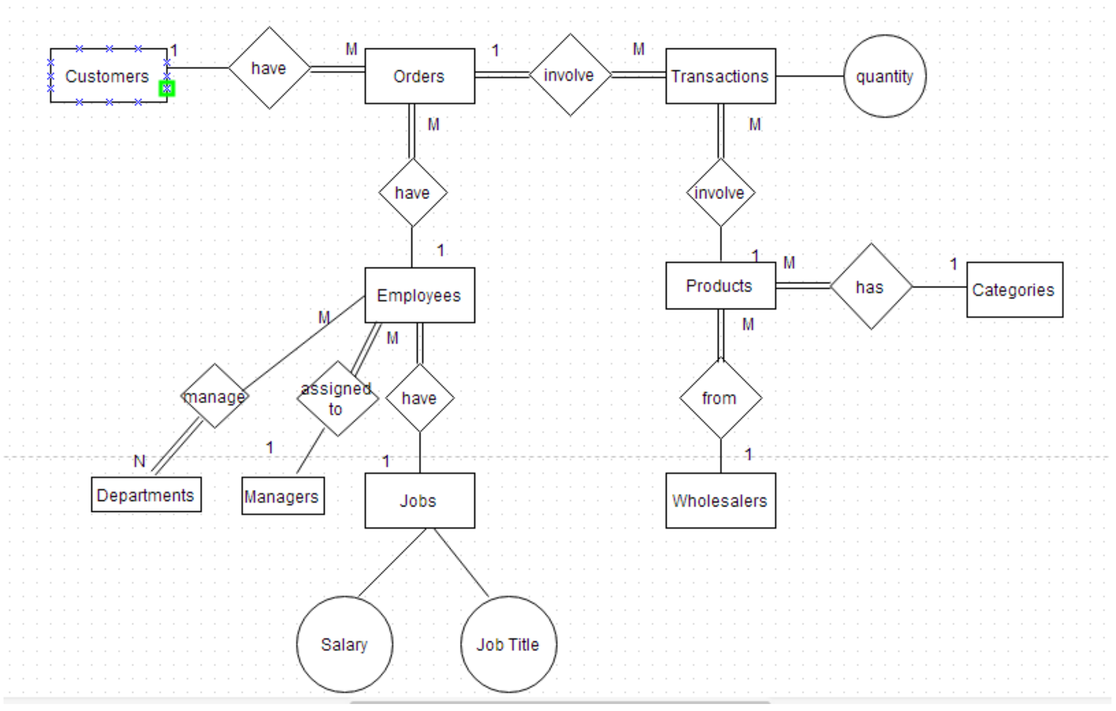







Employees
M
assigned to
1
Managers
Employees
M
manage
N
Departments
Orders
1
involve
M
Transactions
Transactions
M
involve
1
Products
Products
M
from
1
Wholesaler
Products
M
has
1
Categories

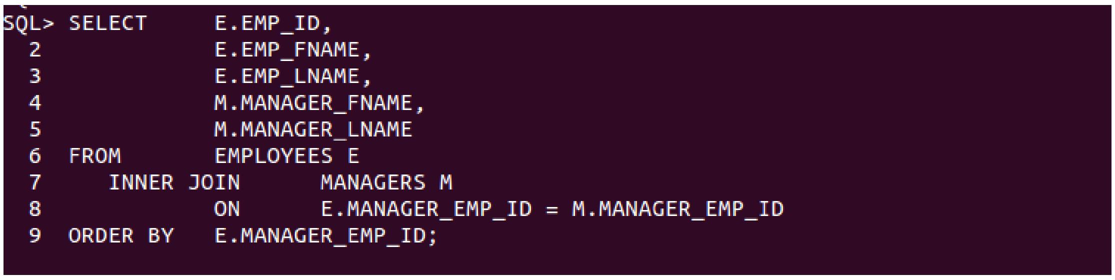
Prototype Entity-Relationship Diagram



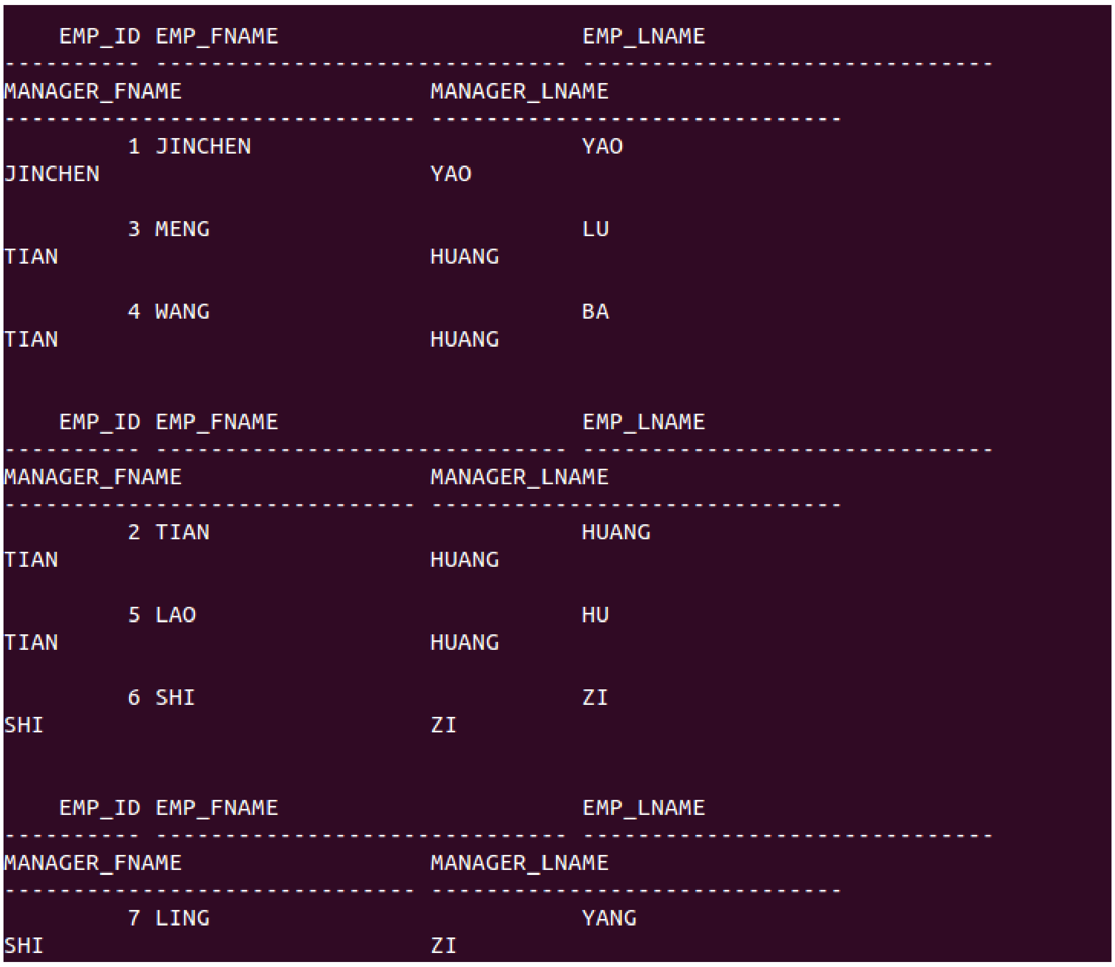
4 INNER JOIN queries with descriptions

/* INNER JOIN 1: List of each employee's name alongside their manager */

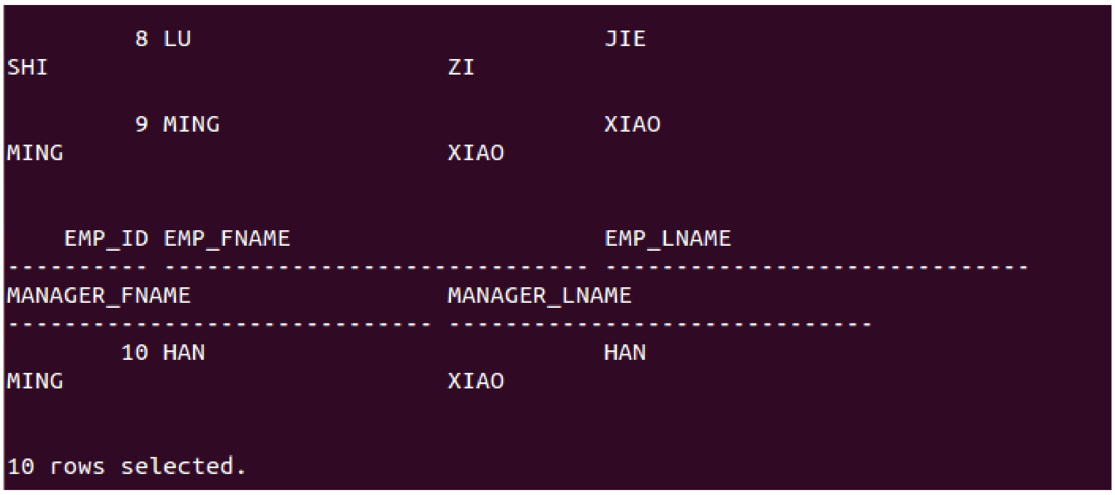
```
SELECT      E.EMP_ID,
            E.EMP_FNAME,
            E.EMP_LNAME,
            M.MANAGER_FNAME,
            M.MANAGER_LNAME
FROM   EMPLOYEES E
      INNER JOIN     MANAGERS M
            ON E.MANAGER_EMP_ID = M.MANAGER_EMP_ID
ORDER BY E.MANAGER_EMP_ID;
```



```
SQL> SELECT     E.EMP_ID,
  2             E.EMP_FNAME,
  3             E.EMP_LNAME,
  4             M.MANAGER_FNAME,
  5             M.MANAGER_LNAME
  6  FROM       EMPLOYEES E
  7     INNER JOIN      MANAGERS M
  8             ON      E.MANAGER_EMP_ID = M.MANAGER_EMP_ID
  9  ORDER BY   E.MANAGER_EMP_ID;
```



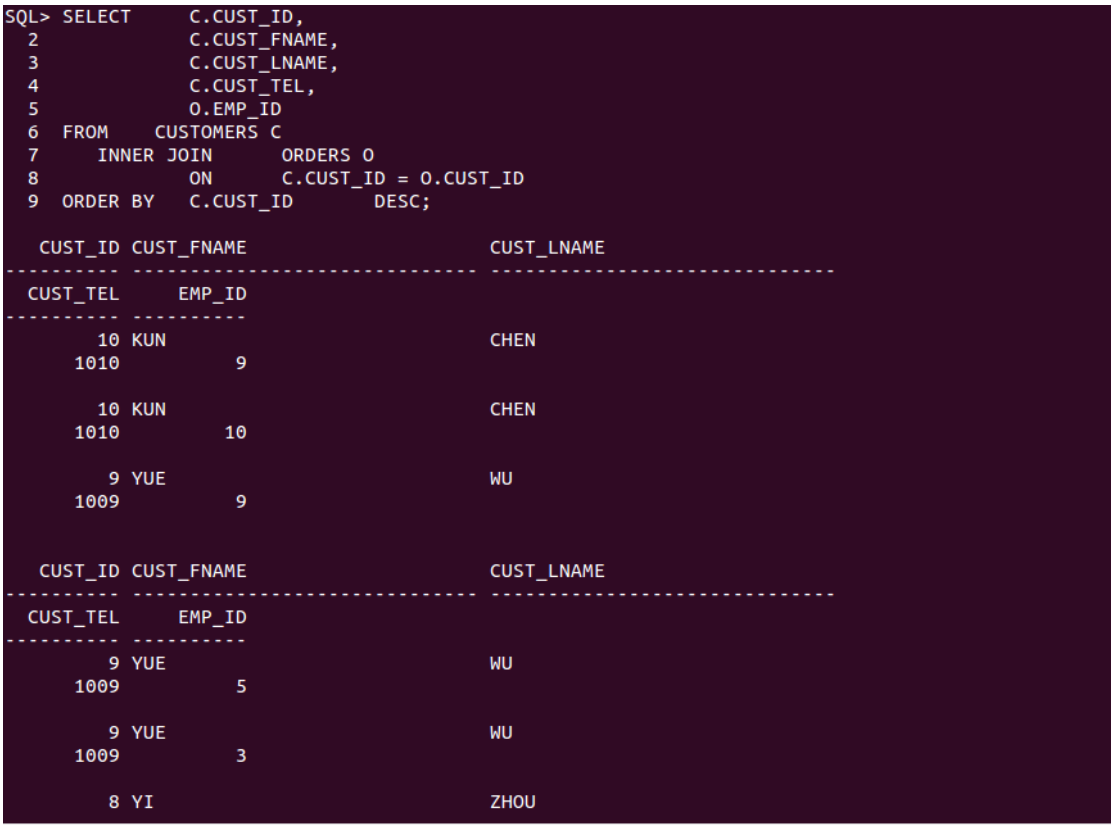
```
    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
MANAGER_FNAME                  MANAGER_LNAME
------------------------------ ------------------------------
         1 JINCHEN                        YAO
JINCHEN                        YAO

         3 MENG                           LU
TIAN                           HUANG

         4 WANG                           BA
TIAN                           HUANG


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
MANAGER_FNAME                  MANAGER_LNAME
------------------------------ ------------------------------
         2 TIAN                           HUANG
TIAN                           HUANG

         5 LAO                            HU
TIAN                           HUANG

         6 SHI                            ZI
SHI                            ZI


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
MANAGER_FNAME                  MANAGER_LNAME
------------------------------ ------------------------------
         7 LING                           YANG
SHI                            ZI
```



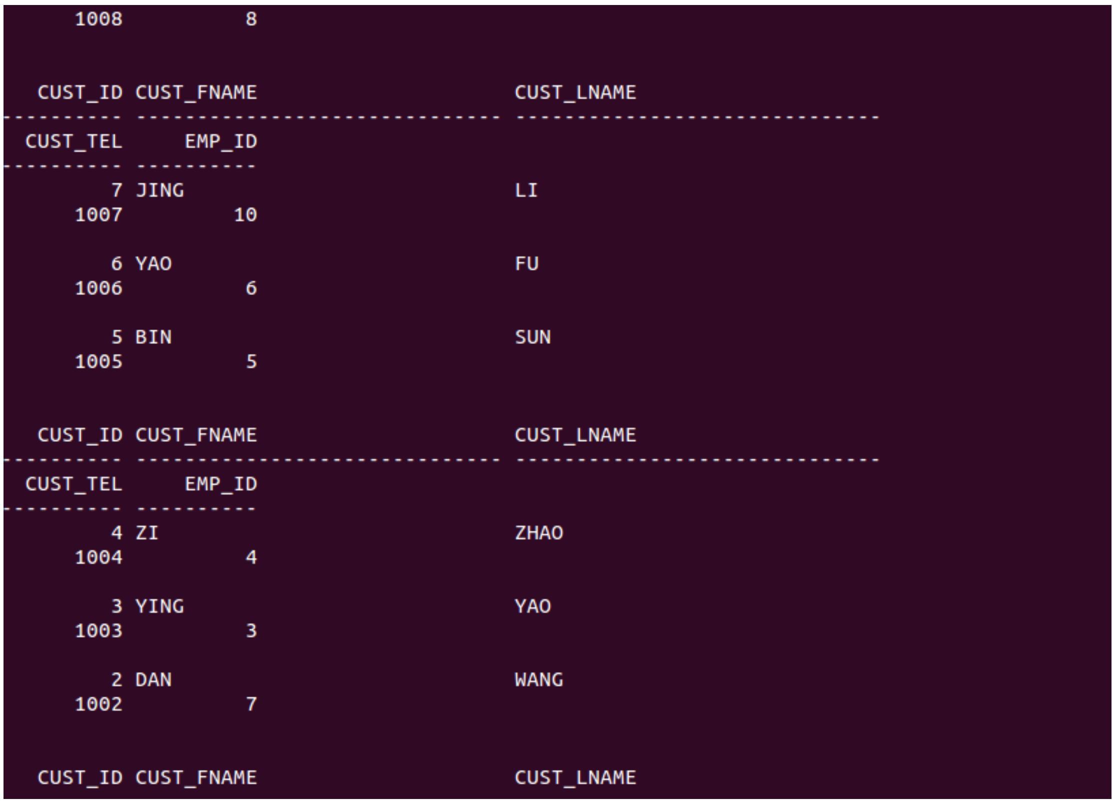
```
         8 LU                             JIE
SHI                            ZI

         9 MING                           XIAO
MING                           XIAO


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
MANAGER_FNAME                  MANAGER_LNAME
------------------------------ ------------------------------
        10 HAN                            HAN
MING                           XIAO


10 rows selected.
```

/* INNER JOIN 2: List of each customer's name alongside the employee they make orders with */

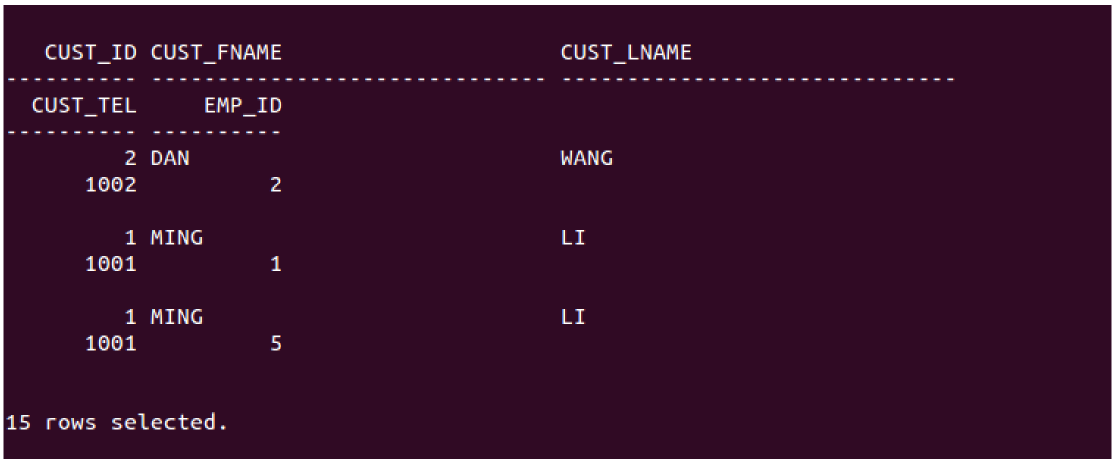
```
SELECT      C.CUST_ID
            C.CUST_FNAME,
            C.CUST_LNAME,
            C.CUST_TEL,
            R.EMP_ID,
FROM        CUSTOMERS C
      INNER JOIN      RELATIONSHIPS R
            ON        C.CUST_ID = O.CUST_ID
ORDER BY C.CUST_ID;
```



```
SQL> SELECT     C.CUST_ID,
               C.CUST_FNAME,
               C.CUST_LNAME,
               C.CUST_TEL,
               O.EMP_ID
     FROM    CUSTOMERS C
        INNER JOIN      ORDERS O
               ON      C.CUST_ID = O.CUST_ID
     ORDER BY   C.CUST_ID       DESC;

   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  CUST_TEL     EMP_ID
---------- ----------
        10 KUN                            CHEN
      1010          9

        10 KUN                            CHEN
      1010         10

         9 YUE                            WU
      1009          9


   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  CUST_TEL     EMP_ID
---------- ----------
         9 YUE                            WU
      1009          5

         9 YUE                            WU
      1009          3

         8 YI                             ZHOU
```



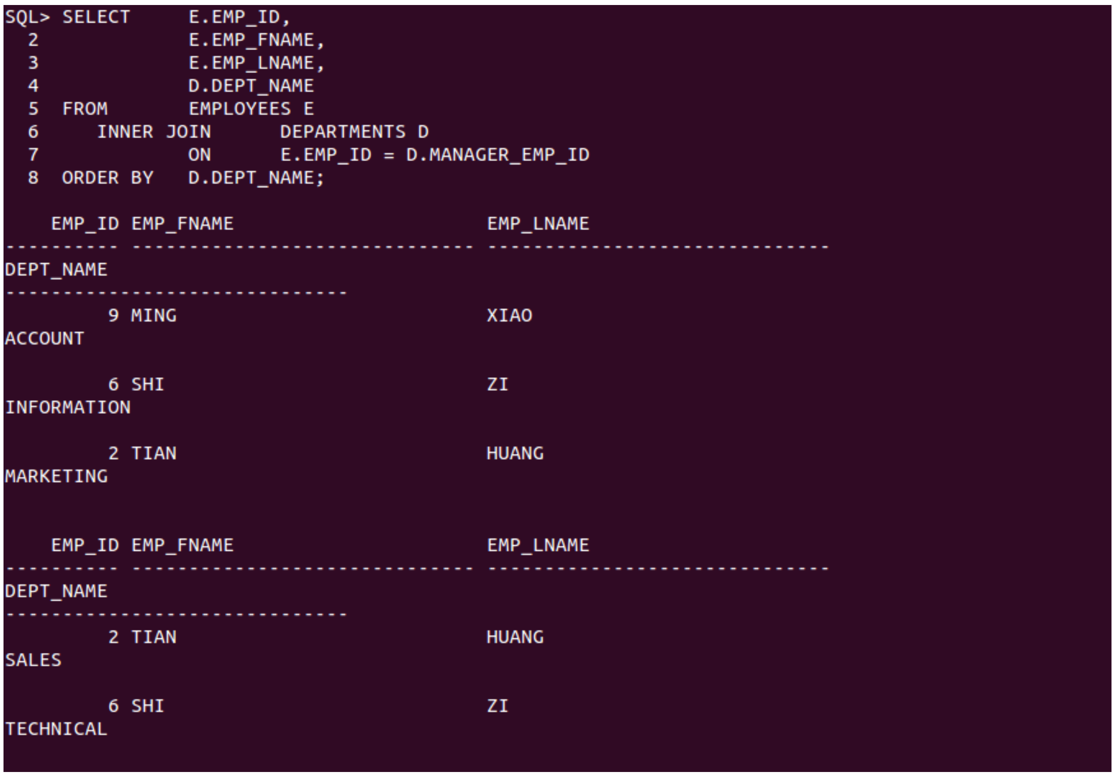
```
      1008          8


   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  CUST_TEL     EMP_ID
---------- ----------
         7 JING                           LI
      1007         10

         6 YAO                            FU
      1006          6

         5 BIN                            SUN
      1005          5


   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  CUST_TEL     EMP_ID
---------- ----------
         4 ZI                             ZHAO
      1004          4

         3 YING                           YAO
      1003          3

         2 DAN                            WANG
      1002          7


   CUST_ID CUST_FNAME                     CUST_LNAME
```



```
   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  CUST_TEL     EMP_ID
---------- ----------
         2 DAN                            WANG
      1002          2

         1 MING                           LI
      1001          1

         1 MING                           LI
      1001          5


15 rows selected.
```

/* INNER JOIN 3: List of each department alongside the manager of the department. */

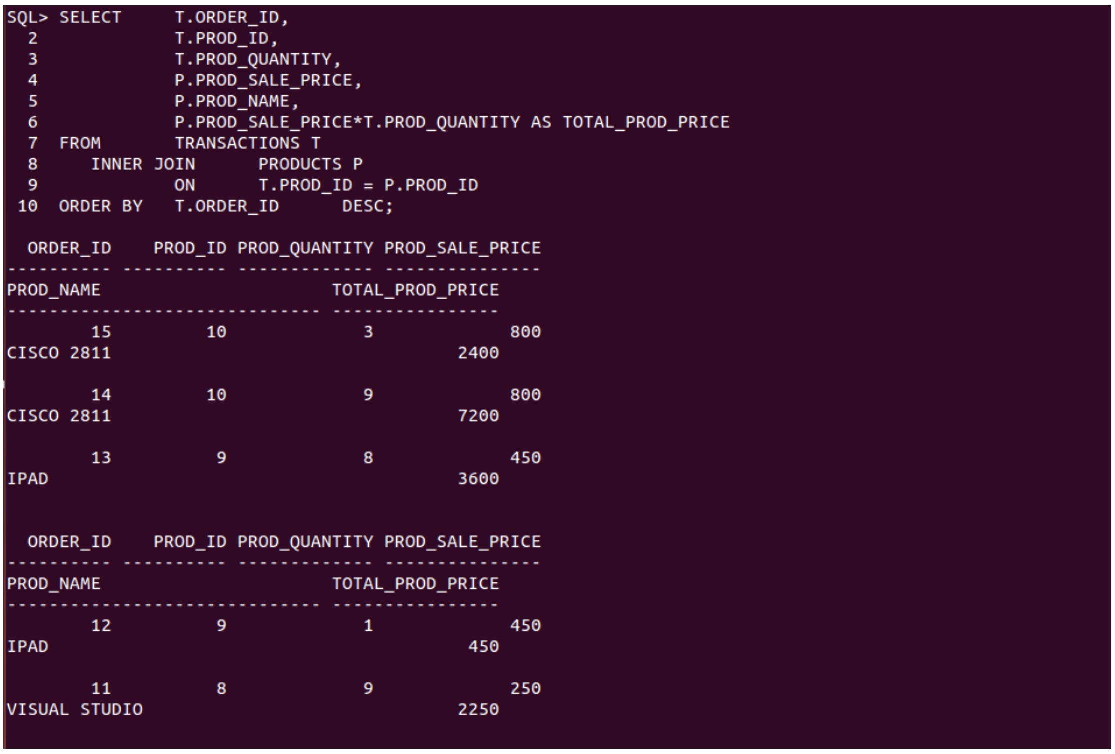
```
SELECT      E.EMP_ID,
            E.EMP_FNAME,
            E.EMP_LNAME,
            D.DEPT_NAME
FROM    EMPLOYEES E
      INNER JOIN      DEPARTMENTS D
            ON        E.EMP_ID = D.MANAGER_EMP_ID
ORDER BY D.DEPT_NAME;
```



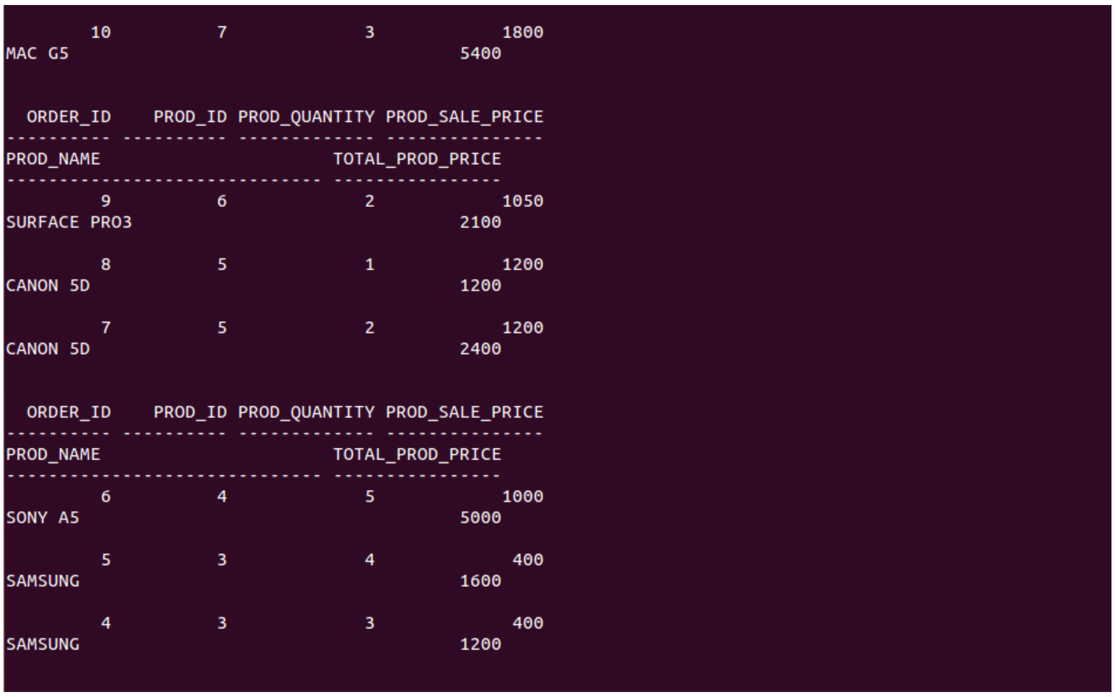
```
SQL> SELECT     E.EMP_ID,
  2             E.EMP_FNAME,
  3             E.EMP_LNAME,
  4             D.DEPT_NAME
  5  FROM       EMPLOYEES E
  6     INNER JOIN      DEPARTMENTS D
  7             ON      E.EMP_ID = D.MANAGER_EMP_ID
  8  ORDER BY   D.DEPT_NAME;

    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
DEPT_NAME
------------------------------
         9 MING                           XIAO
ACCOUNT

         6 SHI                            ZI
INFORMATION

         2 TIAN                           HUANG
MARKETING


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
DEPT_NAME
------------------------------
         2 TIAN                           HUANG
SALES

         6 SHI                            ZI
TECHNICAL
```

/* INNER JOIN 4: List of each order id, starting with most recent order, alongside any product and quantities in each order. The total product price in each order is shown which is the quantity multiplied by the product sale price. */

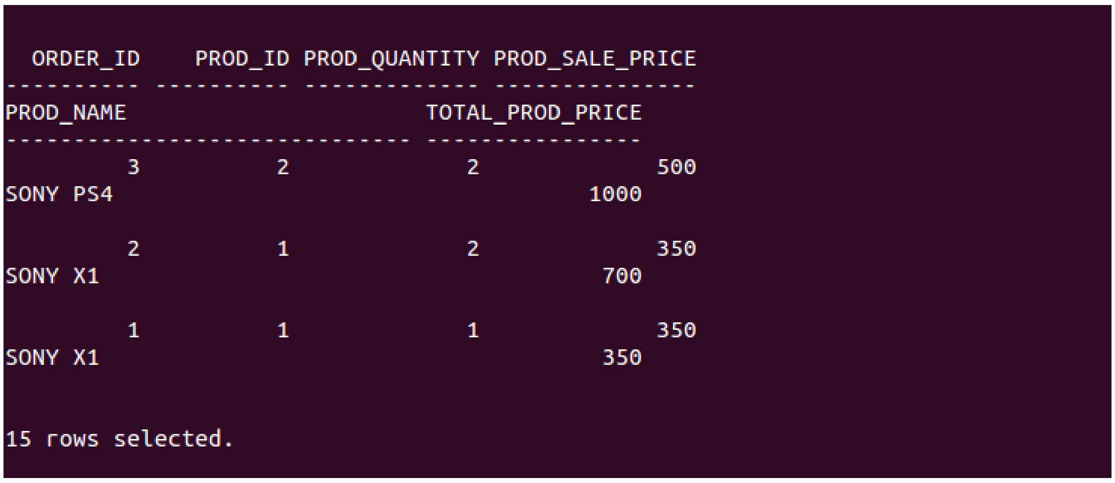
```
SELECT     T.ORDER_ID,
           T.PROD_ID,
           T.PROD_QUANTITY,
           P.PROD_SALE_PRICE,
           P.PROD_NAME,
           P.PROD_SALE_PRICE*T.PROD_QUANTITY AS TOTAL_PROD_PRICE
FROM       TRANSACTIONS T
     INNER JOIN     PRODUCTS P
           ON T.PROD_ID = P.PROD_ID
ORDER BY T.ORDER_ID    DESC;
```



```
SQL> SELECT     T.ORDER_ID,
  2             T.PROD_ID,
  3             T.PROD_QUANTITY,
  4             P.PROD_SALE_PRICE,
  5             P.PROD_NAME,
  6             P.PROD_SALE_PRICE*T.PROD_QUANTITY AS TOTAL_PROD_PRICE
  7  FROM       TRANSACTIONS T
  8     INNER JOIN      PRODUCTS P
  9             ON      T.PROD_ID = P.PROD_ID
 10  ORDER BY   T.ORDER_ID      DESC;

  ORDER_ID    PROD_ID PROD_QUANTITY PROD_SALE_PRICE
---------- ---------- ------------- --------------
PROD_NAME                      TOTAL_PROD_PRICE
------------------------------ ----------------
        15         10             3            800
CISCO 2811                                 2400

        14         10             9            800
CISCO 2811                                 7200

        13          9             8            450
IPAD                                       3600


  ORDER_ID    PROD_ID PROD_QUANTITY PROD_SALE_PRICE
---------- ---------- ------------- --------------
PROD_NAME                      TOTAL_PROD_PRICE
------------------------------ ----------------
        12          9             1            450
IPAD                                        450

        11          8             9            250
VISUAL STUDIO                              2250
```



```
        10          7             3           1800
MAC G5                                     5400


  ORDER_ID    PROD_ID PROD_QUANTITY PROD_SALE_PRICE
---------- ---------- ------------- --------------
PROD_NAME                      TOTAL_PROD_PRICE
------------------------------ ----------------
         9          6             2           1050
SURFACE PRO3                               2100

         8          5             1           1200
CANON 5D                                   1200

         7          5             2           1200
CANON 5D                                   2400


  ORDER_ID    PROD_ID PROD_QUANTITY PROD_SALE_PRICE
---------- ---------- ------------- --------------
PROD_NAME                      TOTAL_PROD_PRICE
------------------------------ ----------------
         6          4             5           1000
SONY A5                                    5000

         5          3             4            400
SAMSUNG                                    1600

         4          3             3            400
SAMSUNG                                    1200
```

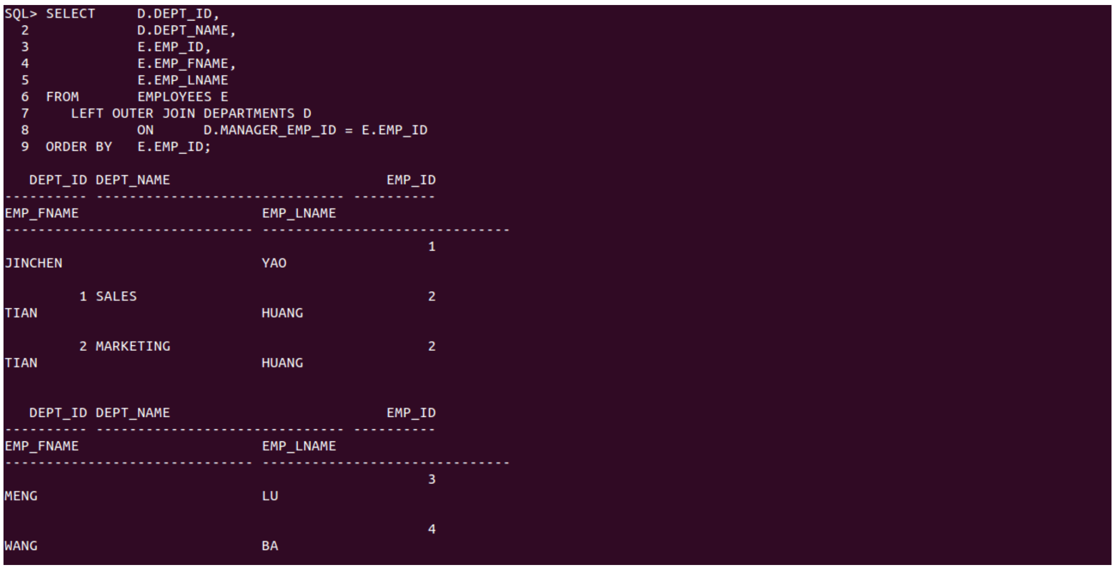


```
  ORDER_ID    PROD_ID PROD_QUANTITY PROD_SALE_PRICE
---------- ---------- ------------- ---------------
PROD_NAME                      TOTAL_PROD_PRICE
------------------------------ ----------------
         3          2             2             500
SONY PS4                                   1000

         2          1             2             350
SONY X1                                     700

         1          1             1             350
SONY X1                                     350


15 rows selected.
```

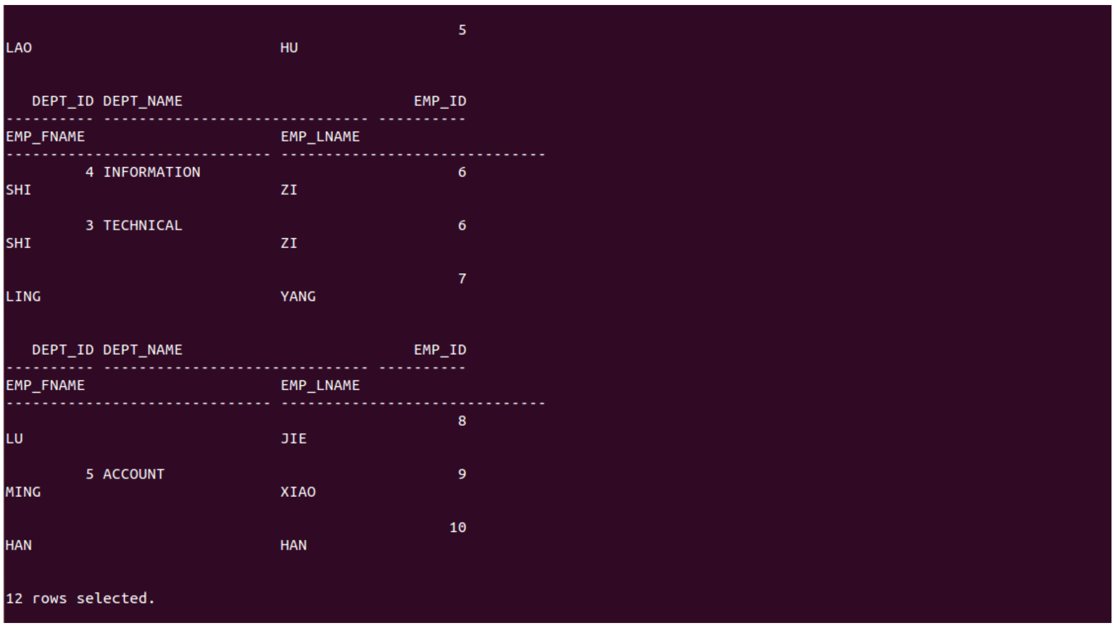
/*   6 OUTER JOIN (2xleft,2xfull,2xright) queries with descriptions   */

/* LEFT OUTER JOIN 1: Show employee id, firstname, lastname and the department that they are the manager of. Also show employee's who are not managers. */

```
SELECT      D.DEPT_ID,
            D.DEPT_NAME,
            E.EMP_ID,
            E.EMP_FNAME,
            E.EMP_LNAME
FROM    EMPLOYEES E
      LEFT OUTER JOIN  DEPARTMENTS D
            ON D.MANAGER_EMP_ID = E.EMP_ID
ORDER BY E.EMP_ID;
```



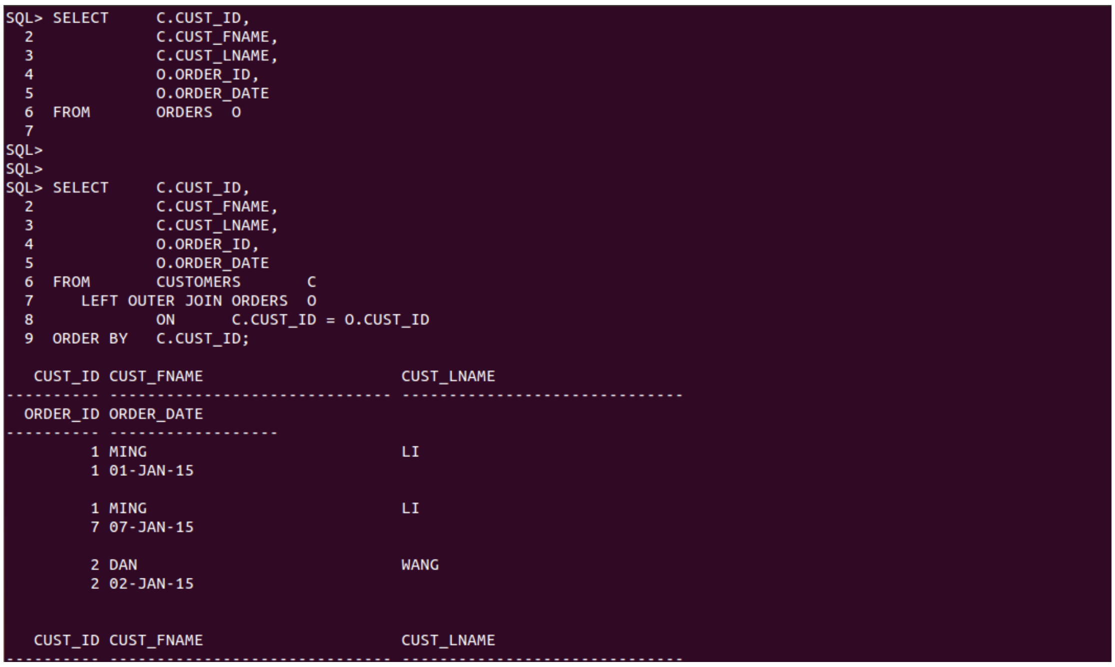
```
SQL> SELECT     D.DEPT_ID,
                D.DEPT_NAME,
                E.EMP_ID,
                E.EMP_FNAME,
                E.EMP_LNAME
     FROM       EMPLOYEES E
        LEFT OUTER JOIN DEPARTMENTS D
                ON      D.MANAGER_EMP_ID = E.EMP_ID
     ORDER BY   E.EMP_ID;

   DEPT_ID DEPT_NAME                          EMP_ID
---------- ------------------------------ ----------
EMP_FNAME                      EMP_LNAME
------------------------------ ------------------------------
                                                   1
JINCHEN                        YAO

         1 SALES                                   2
TIAN                           HUANG

         2 MARKETING                               2
TIAN                           HUANG


   DEPT_ID DEPT_NAME                          EMP_ID
---------- ------------------------------ ----------
EMP_FNAME                      EMP_LNAME
------------------------------ ------------------------------
                                                   3
MENG                           LU

                                                   4
WANG                           BA
```



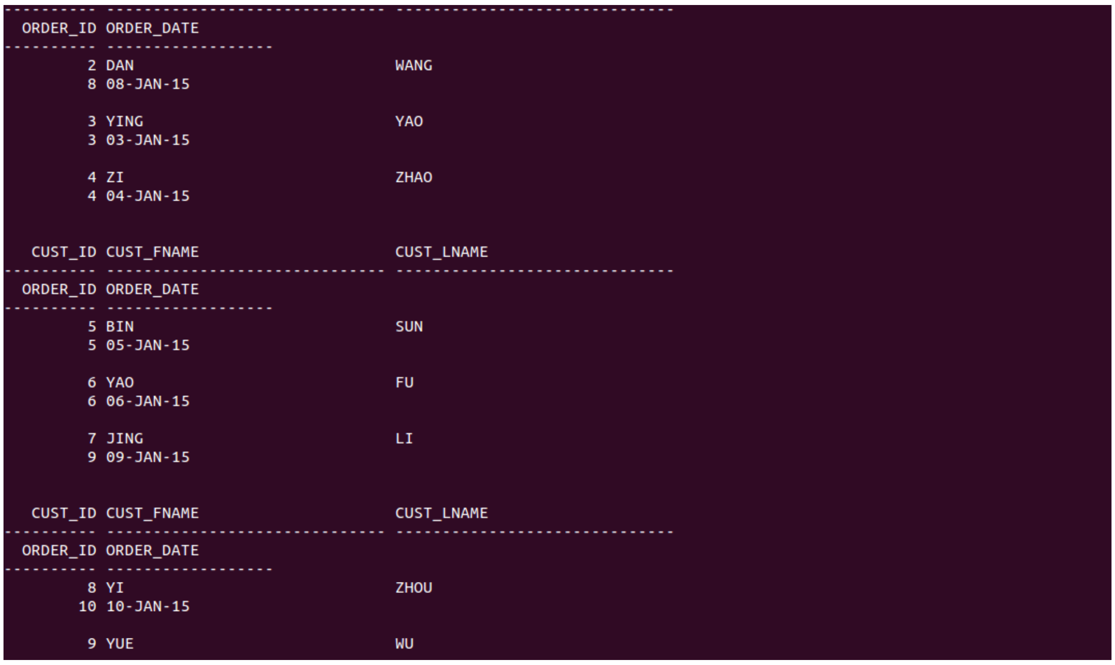
```
                                                   5
LAO                            HU

   DEPT_ID DEPT_NAME                          EMP_ID
---------- ------------------------------ ----------
EMP_FNAME                      EMP_LNAME
------------------------------ ------------------------------
         4 INFORMATION                             6
SHI                            ZI

         3 TECHNICAL                               6
SHI                            ZI

                                                   7
LING                           YANG

   DEPT_ID DEPT_NAME                          EMP_ID
---------- ------------------------------ ----------
EMP_FNAME                      EMP_LNAME
------------------------------ ------------------------------
                                                   8
LU                             JIE

         5 ACCOUNT                                 9
MING                           XIAO

                                                  10
HAN                            HAN

12 rows selected.
```

/* LEFT OUTER JOIN 2: Show all customers id, firstname, lastname, order id, date including customers who haven't ordered anything yet. Show orders based on most recent order. */

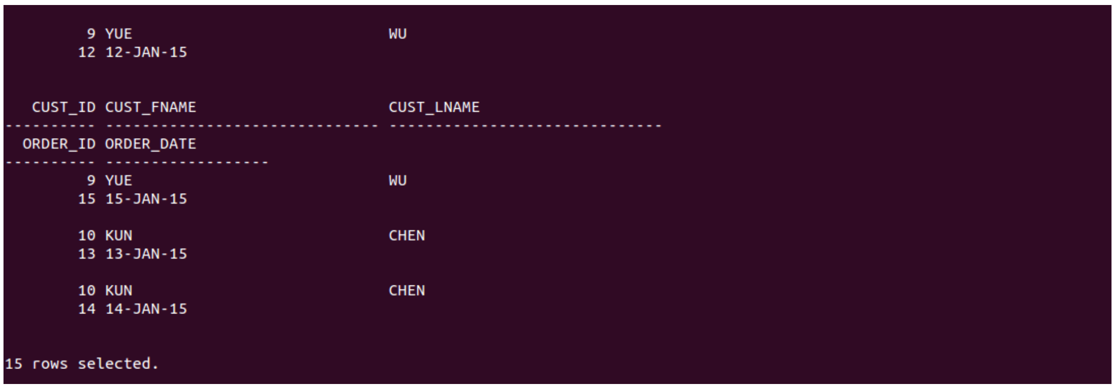
```
SELECT     C.CUST_ID,
           C.CUST_FNAME,
           C.CUST_LNAME,
           O.ORDER_ID,
           O.ORDER_DATE
FROM       CUSTOMERS    C
      LEFT OUTER JOIN ORDERS    O
           ON C.CUST_ID = O.CUST_ID
ORDER BY C.CUST_ID;
```



```
SQL> SELECT     C.CUST_ID,
  2             C.CUST_FNAME,
  3             C.CUST_LNAME,
  4             O.ORDER_ID,
  5             O.ORDER_DATE
  6  FROM       ORDERS  O
  7
SQL>
SQL>
SQL> SELECT     C.CUST_ID,
  2             C.CUST_FNAME,
  3             C.CUST_LNAME,
  4             O.ORDER_ID,
  5             O.ORDER_DATE
  6  FROM       CUSTOMERS      C
  7     LEFT OUTER JOIN ORDERS  O
  8             ON      C.CUST_ID = O.CUST_ID
  9  ORDER BY   C.CUST_ID;

   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID ORDER_DATE
---------- ------------------
         1 MING                           LI
         1 01-JAN-15

         1 MING                           LI
         7 07-JAN-15

         2 DAN                            WANG
         2 02-JAN-15


   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
```



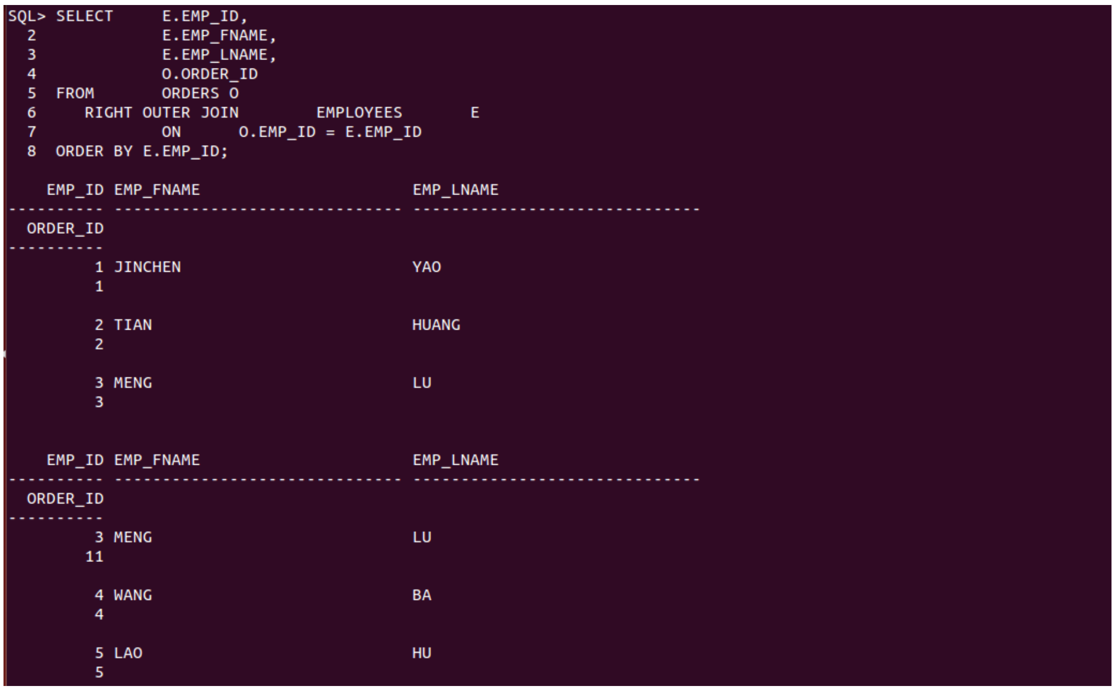
```
---------- ------------------------------ ------------------------------
  ORDER_ID ORDER_DATE
---------- ------------------
         2 DAN                            WANG
         8 08-JAN-15

         3 YING                           YAO
         3 03-JAN-15

         4 ZI                             ZHAO
         4 04-JAN-15


   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID ORDER_DATE
---------- ------------------
         5 BIN                            SUN
         5 05-JAN-15

         6 YAO                            FU
         6 06-JAN-15

         7 JING                           LI
         9 09-JAN-15


   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID ORDER_DATE
---------- ------------------
         8 YI                             ZHOU
        10 10-JAN-15

         9 YUE                            WU
```



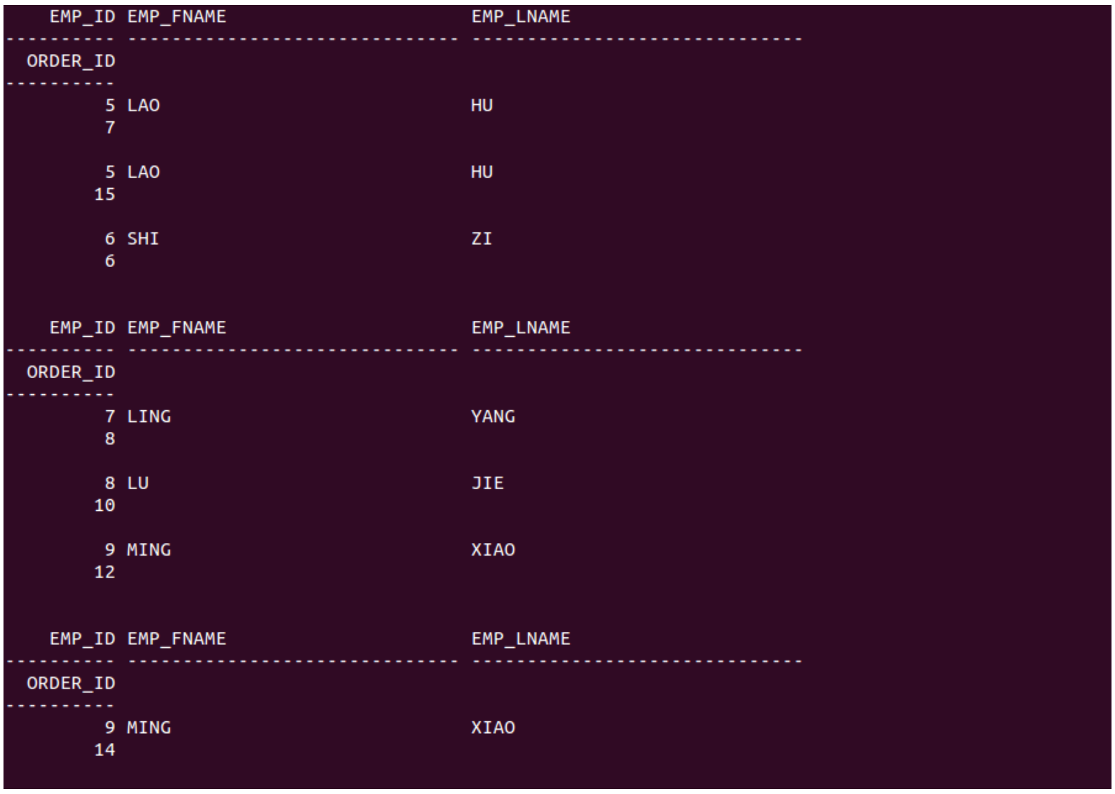
```
         9 YUE                            WU
        12 12-JAN-15


   CUST_ID CUST_FNAME                     CUST_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID ORDER_DATE
---------- ------------------
         9 YUE                            WU
        15 15-JAN-15

        10 KUN                            CHEN
        13 13-JAN-15

        10 KUN                            CHEN
        14 14-JAN-15


15 rows selected.
```

/* RIGHT OUTER JOIN 1: Display employee id, firstname, lastname and the orders that they were involved with. Also show employee's who have not been involved with an order. */

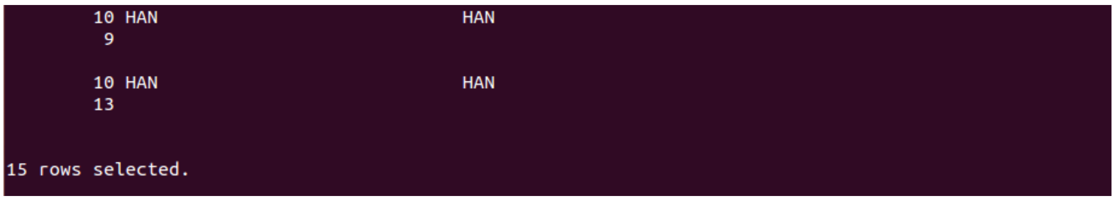
```
SELECT      E.EMP_ID,
            E.EMP_FNAME,
            E.EMP_LNAME,
            O.ORDER_ID
FROM    ORDERS O
      RIGHT OUTER JOIN      EMPLOYEES    E
            ON O.EMP_ID = E.EMP_ID
ORDER BY E.EMP_ID;
```



```
SQL> SELECT      E.EMP_ID,
  2            E.EMP_FNAME,
  3            E.EMP_LNAME,
  4            O.ORDER_ID
  5  FROM      ORDERS O
  6     RIGHT OUTER JOIN        EMPLOYEES       E
  7            ON      O.EMP_ID = E.EMP_ID
  8  ORDER BY E.EMP_ID;

    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID
----------
         1 JINCHEN                        YAO
         1

         2 TIAN                           HUANG
         2

         3 MENG                           LU
         3


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID
----------
         3 MENG                           LU
        11

         4 WANG                           BA
         4

         5 LAO                            HU
         5
```



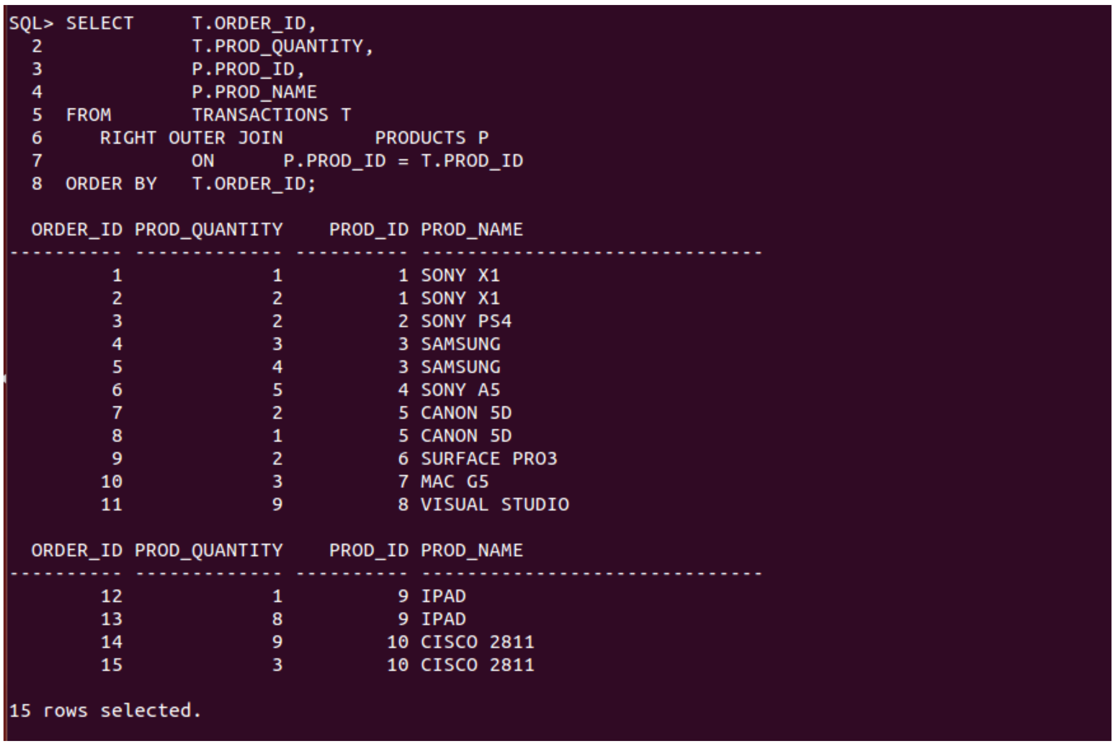
```
    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID
----------
         5 LAO                            HU
         7

         5 LAO                            HU
        15

         6 SHI                            ZI
         6


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID
----------
         7 LING                           YANG
         8

         8 LU                             JIE
        10

         9 MING                           XIAO
        12


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
  ORDER_ID
----------
         9 MING                           XIAO
        14
```



```
        10 HAN                            HAN
         9

        10 HAN                            HAN
        13


15 rows selected.
```

/* RIGHT OUTER JOIN 2: Display all orders, the quantities of products that have been ordered in that order and the product ID and name. Also show products that have never been ordered. */

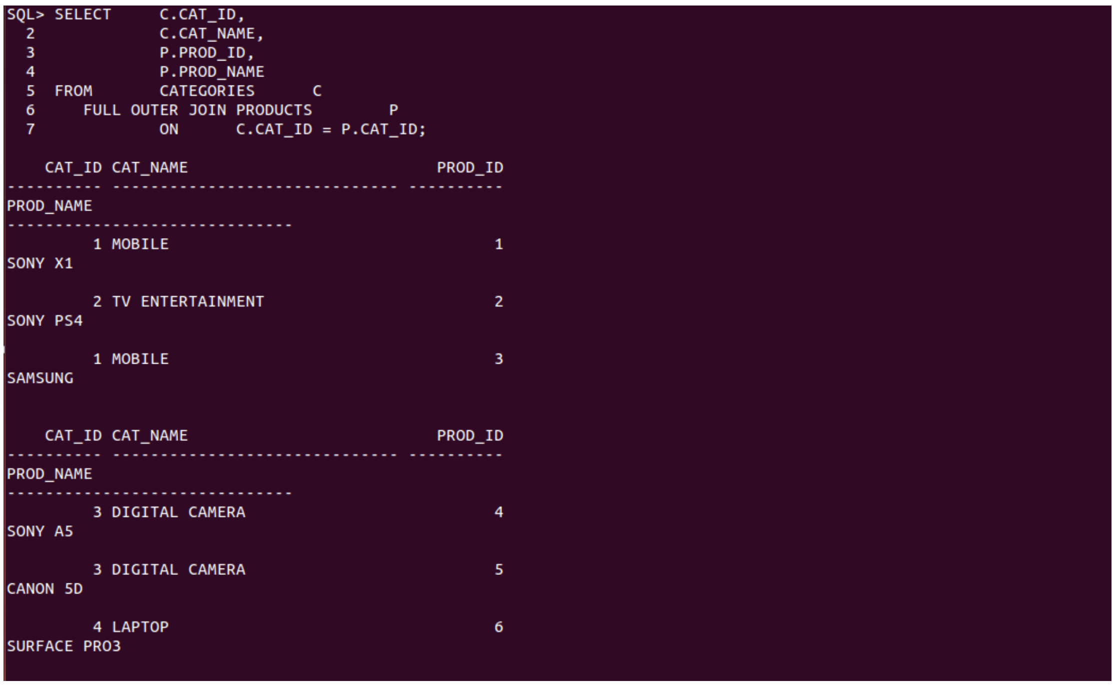
```
SELECT    T.ORDER_ID,
          T.PROD_QUANTITY,
          P.PROD_ID,
          P.PROD_NAME
FROM      TRANSACTIONS T
     RIGHT OUTER JOIN    PRODUCTS P
          ON P.PROD_ID = T.PROD_ID
ORDER BY T.ORDER_ID;
```



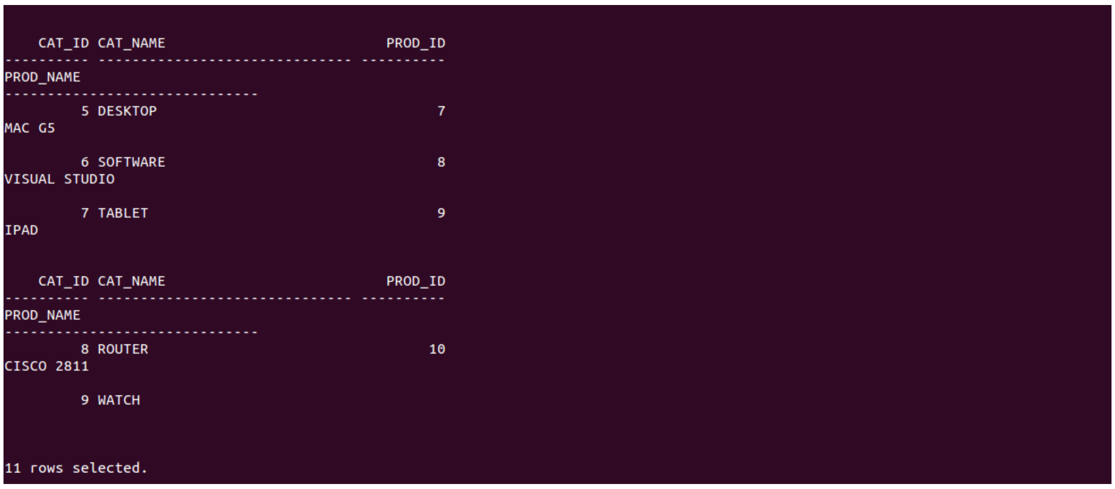
```
SQL> SELECT     T.ORDER_ID,
                T.PROD_QUANTITY,
                P.PROD_ID,
                P.PROD_NAME
     FROM       TRANSACTIONS T
        RIGHT OUTER JOIN        PRODUCTS P
                ON      P.PROD_ID = T.PROD_ID
     ORDER BY   T.ORDER_ID;

  ORDER_ID PROD_QUANTITY    PROD_ID PROD_NAME
---------- ------------- ---------- ------------------------------
         1             1          1 SONY X1
         2             2          1 SONY X1
         3             2          2 SONY PS4
         4             3          3 SAMSUNG
         5             4          3 SAMSUNG
         6             5          4 SONY A5
         7             2          5 CANON 5D
         8             1          5 CANON 5D
         9             2          6 SURFACE PRO3
        10             3          7 MAC G5
        11             9          8 VISUAL STUDIO

  ORDER_ID PROD_QUANTITY    PROD_ID PROD_NAME
---------- ------------- ---------- ------------------------------
        12             1          9 IPAD
        13             8          9 IPAD
        14             9         10 CISCO 2811
        15             3         10 CISCO 2811

15 rows selected.
```

/* FULL OUTER JOIN 1: Show all product names and product cateogries. Show products that have not been assigned a category. */

```
SELECT     C.CAT_ID,
           C.CAT_NAME,
           P.PROD_ID,
           P.PROD_NAME
FROM   CATEGORIES   C
     FULL OUTER JOIN PRODUCTSP
           ON C.CAT_ID = P.CAT_ID;
```



```
SQL> SELECT     C.CAT_ID,
  2             C.CAT_NAME,
  3             P.PROD_ID,
  4             P.PROD_NAME
  5  FROM       CATEGORIES      C
  6     FULL OUTER JOIN PRODUCTS        P
  7             ON      C.CAT_ID = P.CAT_ID;

    CAT_ID CAT_NAME                          PROD_ID
---------- ---------------------------- ----------
PROD_NAME
------------------------------
         1 MOBILE                                1
SONY X1

         2 TV ENTERTAINMENT                      2
SONY PS4

         1 MOBILE                                3
SAMSUNG


    CAT_ID CAT_NAME                          PROD_ID
---------- ---------------------------- ----------
PROD_NAME
------------------------------
         3 DIGITAL CAMERA                        4
SONY A5

         3 DIGITAL CAMERA                        5
CANON 5D

         4 LAPTOP                                6
SURFACE PRO3
```

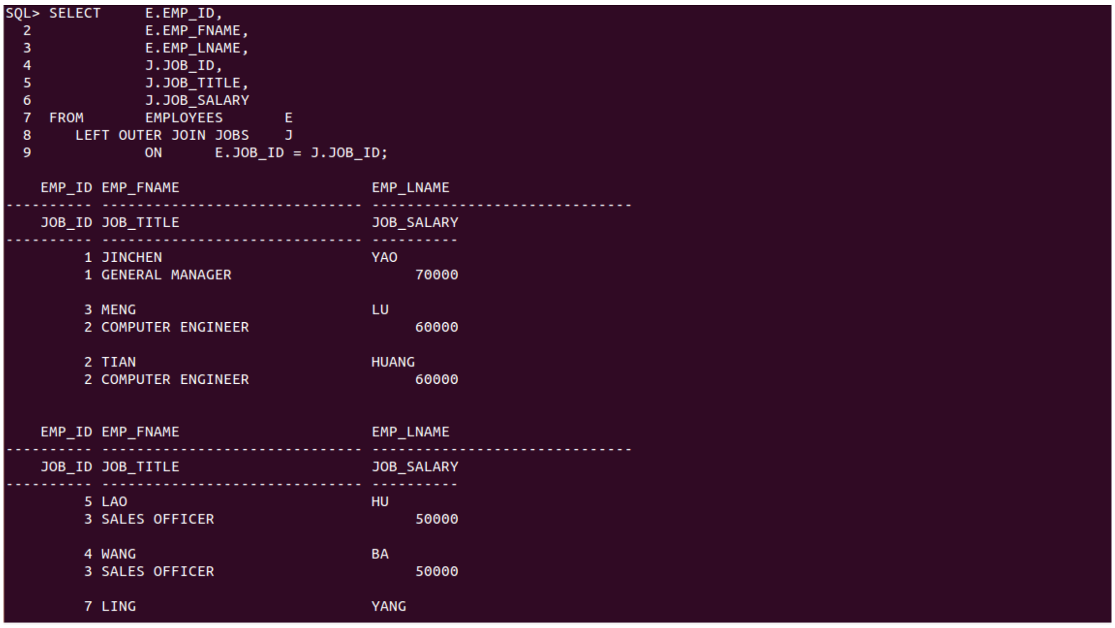


```
    CAT_ID CAT_NAME                          PROD_ID
---------- ---------------------------- ----------
PROD_NAME
------------------------------
         5 DESKTOP                               7
MAC G5

         6 SOFTWARE                              8
VISUAL STUDIO

         7 TABLET                                9
IPAD

    CAT_ID CAT_NAME                          PROD_ID
---------- ---------------------------- ----------
PROD_NAME
------------------------------
         8 ROUTER                               10
CISCO 2811

         9 WATCH


11 rows selected.
```

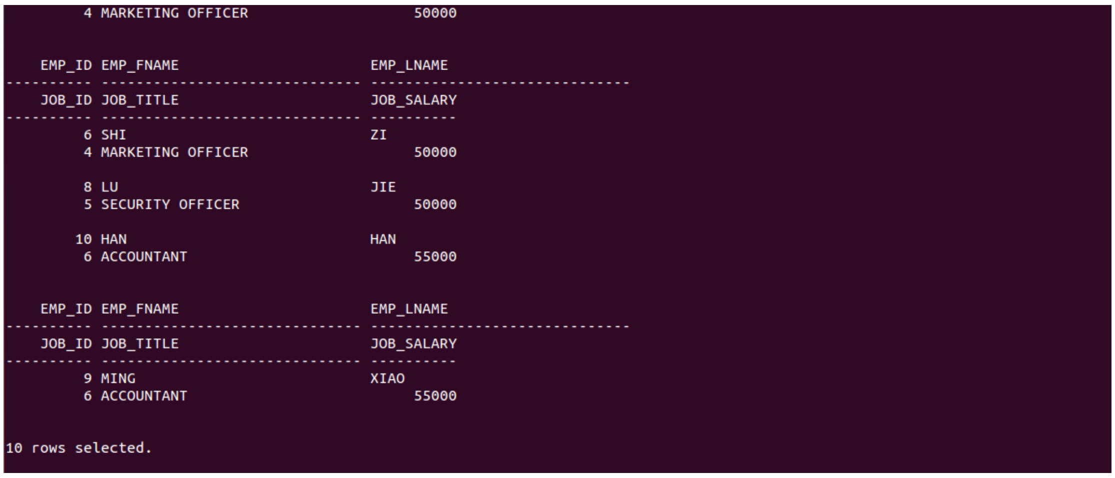
/* FULL OUTER JOIN 2: Display all job and employees. Show jobs that have no employees assigned to them and show all employees even if they have no job assigned to them. Because every employee is required to be assigned a job when being entered into the database, so all employees will be shown alongside their job and there will be no employees without a job. */

```
SELECT      E.EMP_ID,
            E.EMP_FNAME,
            E.EMP_LNAME,
            J.JOB_ID,
            J.JOB_TITLE,
            J.JOB_SALARY
FROM    EMPLOYEES     E
```

```
        LEFT OUTER JOIN  JOBS      J
              ON  E.JOB_ID = J.JOB_ID;
```



```
SQL> SELECT     E.EMP_ID,
  2            E.EMP_FNAME,
  3            E.EMP_LNAME,
  4            J.JOB_ID,
  5            J.JOB_TITLE,
  6            J.JOB_SALARY
  7  FROM       EMPLOYEES       E
  8     LEFT OUTER JOIN JOBS    J
  9            ON      E.JOB_ID = J.JOB_ID;

    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
    JOB_ID JOB_TITLE                      JOB_SALARY
---------- ------------------------------ ----------
         1 JINCHEN                        YAO
         1 GENERAL MANAGER                     70000

         3 MENG                           LU
         2 COMPUTER ENGINEER                   60000

         2 TIAN                           HUANG
         2 COMPUTER ENGINEER                   60000


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
    JOB_ID JOB_TITLE                      JOB_SALARY
---------- ------------------------------ ----------
         5 LAO                            HU
         3 SALES OFFICER                       50000

         4 WANG                           BA
         3 SALES OFFICER                       50000

         7 LING                           YANG
```

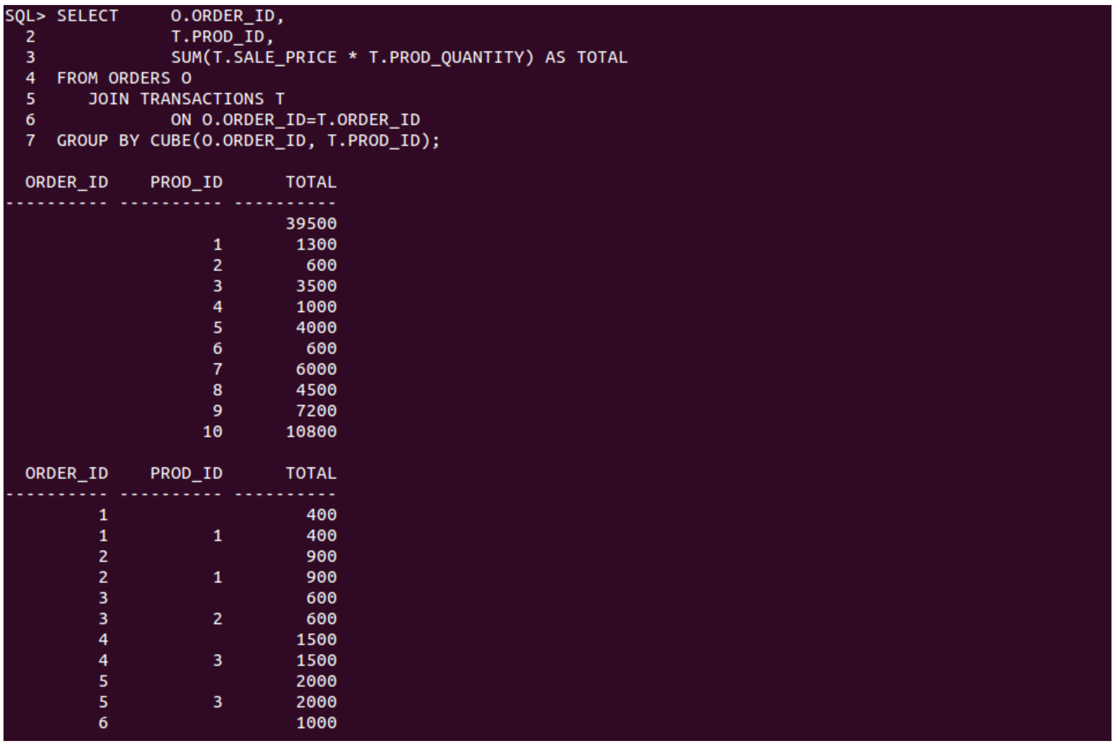


```
         4 MARKETING OFFICER                   50000


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
    JOB_ID JOB_TITLE                      JOB_SALARY
---------- ------------------------------ ----------
         6 SHI                            ZI
         4 MARKETING OFFICER                   50000

         8 LU                             JIE
         5 SECURITY OFFICER                    50000

        10 HAN                            HAN
         6 ACCOUNTANT                          55000


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
    JOB_ID JOB_TITLE                      JOB_SALARY
---------- ------------------------------ ----------
         9 MING                           XIAO
         6 ACCOUNTANT                          55000


10 rows selected.
```

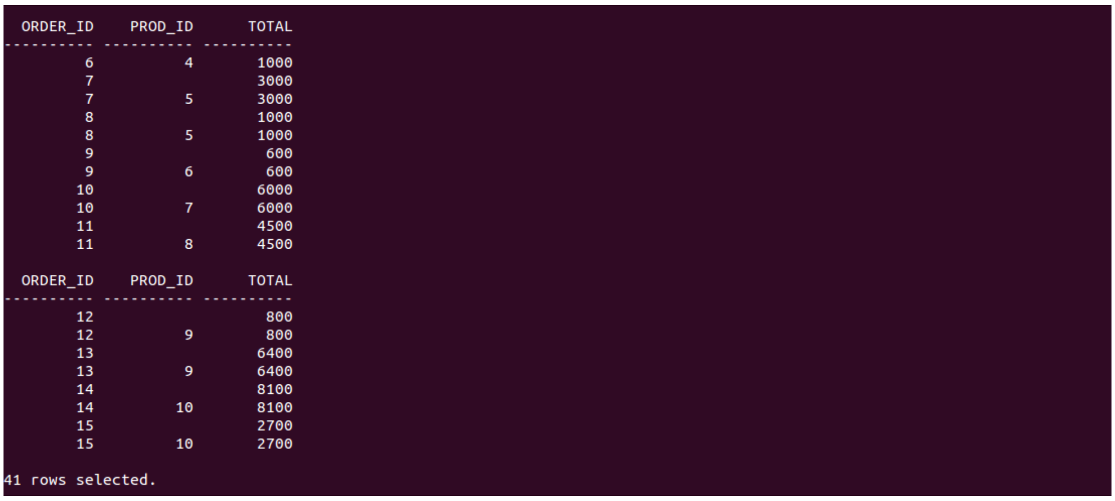
/* 1 CUBE query (with at least 2 columns) */

/* Show each order ID, Product ID, the sum of each order and the sum of all orders of a particular product to show which product is selling the most. Also output total sales. */

```
SELECT      O.ORDER_ID,
            T.PROD_ID,
            SUM(T.SALE_PRICE * T.PROD_QUANTITY) AS TOTAL
FROM ORDERS O
      JOIN TRANSACTIONS T
            ON O.ORDER_ID=T.ORDER_ID
GROUP BY CUBE(O.ORDER_ID, T.PROD_ID);
```



```
SQL> SELECT      O.ORDER_ID,
                 T.PROD_ID,
                 SUM(T.SALE_PRICE * T.PROD_QUANTITY) AS TOTAL
     FROM ORDERS O
        JOIN TRANSACTIONS T
                 ON O.ORDER_ID=T.ORDER_ID
     GROUP BY CUBE(O.ORDER_ID, T.PROD_ID);

  ORDER_ID    PROD_ID      TOTAL
---------- ---------- ----------
                           39500
                    1       1300
                    2        600
                    3       3500
                    4       1000
                    5       4000
                    6        600
                    7       6000
                    8       4500
                    9       7200
                   10      10800

  ORDER_ID    PROD_ID      TOTAL
---------- ---------- ----------
         1                   400
         1          1        400
         2                   900
         2          1        900
         3                   600
         3          2        600
         4                  1500
         4          3       1500
         5                  2000
         5          3       2000
         6                  1000
```

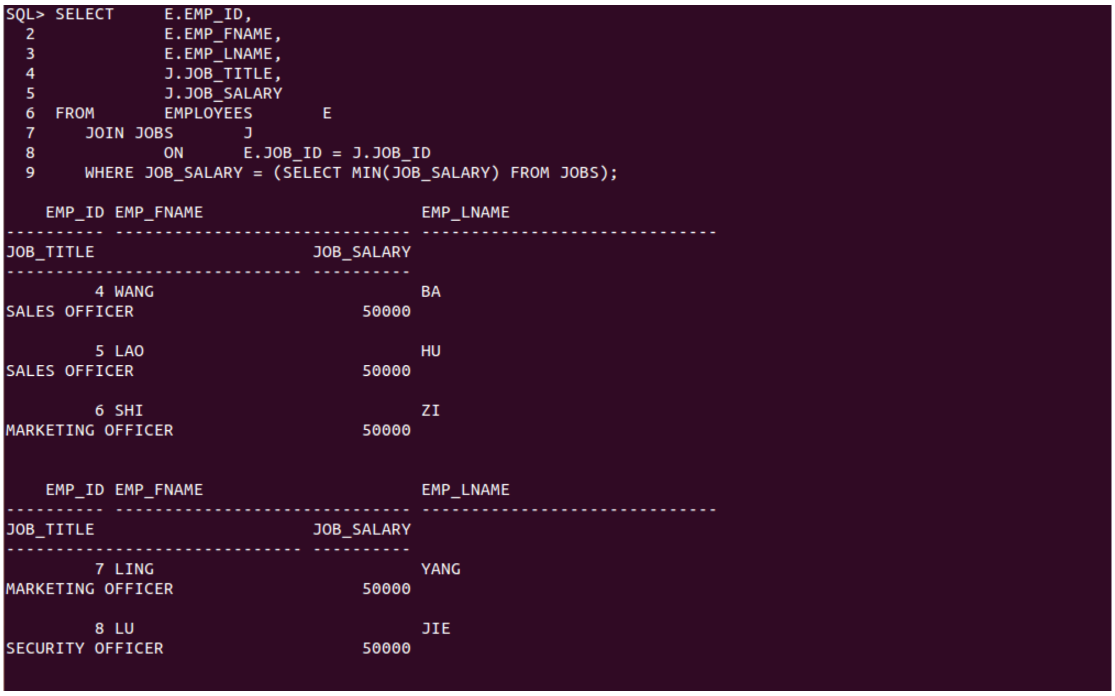


```
  ORDER_ID    PROD_ID      TOTAL
---------- ---------- ----------
         6          4       1000
         7                  3000
         7          5       3000
         8                  1000
         8          5       1000
         9                   600
         9          6        600
        10                  6000
        10          7       6000
        11                  4500
        11          8       4500

  ORDER_ID    PROD_ID      TOTAL
---------- ---------- ----------
        12                   800
        12          9        800
        13                  6400
        13          9       6400
        14                  8100
        14         10       8100
        15                  2700
        15         10       2700

41 rows selected.
```

/*   5 examples of subqueries   */

/* Sub-query 1: Show the employees who are have the lowest salary. */

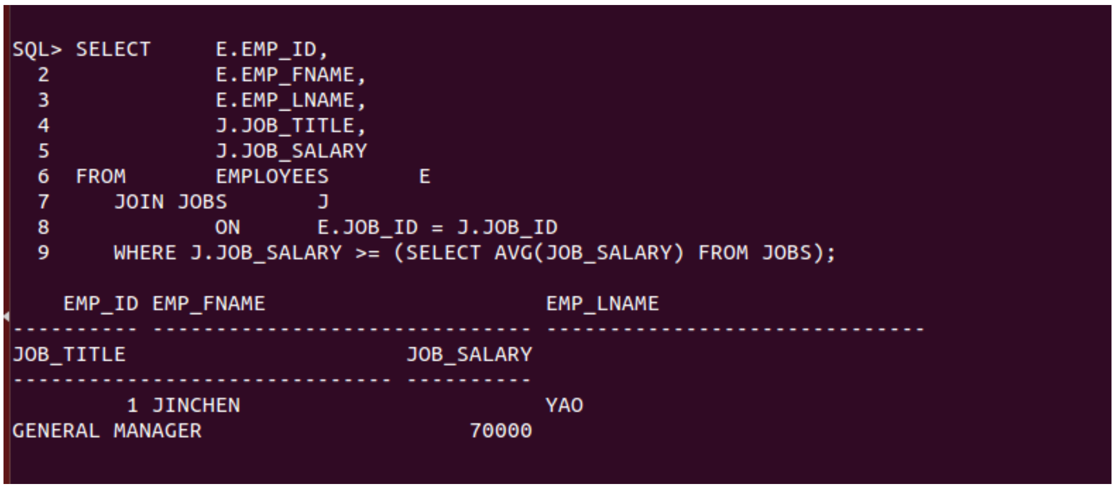
```
SELECT      E.EMP_ID,
            E.EMP_FNAME,
            E.EMP_LNAME,
            J.JOB_TITLE,
            J.JOB_SALARY
FROM    EMPLOYEES     E
      JOIN JOBS         J
            ON        E.JOB_ID = J.JOB_ID
      WHERE JOB_SALARY = (SELECT MIN(JOB_SALARY) FROM JOBS);
```



```
SQL> SELECT     E.EMP_ID,
                E.EMP_FNAME,
                E.EMP_LNAME,
                J.JOB_TITLE,
                J.JOB_SALARY
     FROM       EMPLOYEES       E
        JOIN JOBS       J
                ON      E.JOB_ID = J.JOB_ID
        WHERE JOB_SALARY = (SELECT MIN(JOB_SALARY) FROM JOBS);

    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
JOB_TITLE                      JOB_SALARY
------------------------------ ----------
         4 WANG                           BA
SALES OFFICER                       50000

         5 LAO                            HU
SALES OFFICER                       50000

         6 SHI                            ZI
MARKETING OFFICER                   50000


    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
JOB_TITLE                      JOB_SALARY
------------------------------ ----------
         7 LING                           YANG
MARKETING OFFICER                   50000

         8 LU                             JIE
SECURITY OFFICER                    50000
```

/* Sub-query 2: Show employees who are on a salary that is more than the average wage in the company. */

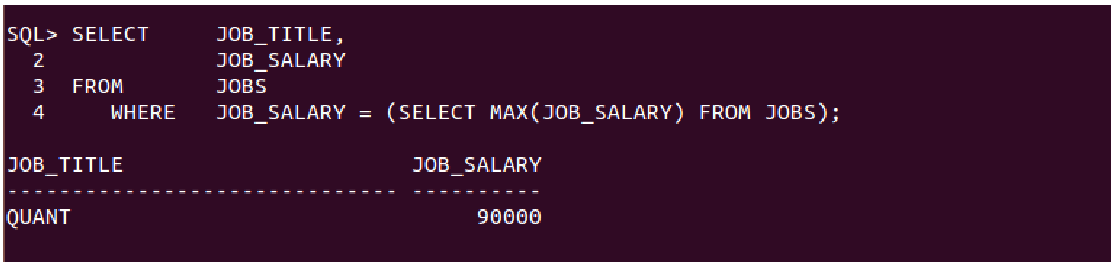
```
SELECT      E.EMP_ID,
            E.EMP_FNAME,
            E.EMP_LNAME,
            J.JOB_TITLE,
            J.JOB_SALARY
FROM    EMPLOYEES     E
      JOIN JOBS         J
            ON        E.JOB_ID = J.JOB_ID
      WHERE J.JOB_SALARY >= (SELECT AVG(JOB_SALARY) FROM JOBS);
```



```
SQL> SELECT     E.EMP_ID,
  2             E.EMP_FNAME,
  3             E.EMP_LNAME,
  4             J.JOB_TITLE,
  5             J.JOB_SALARY
  6  FROM       EMPLOYEES       E
  7     JOIN JOBS       J
  8             ON      E.JOB_ID = J.JOB_ID
  9     WHERE J.JOB_SALARY >= (SELECT AVG(JOB_SALARY) FROM JOBS);

    EMP_ID EMP_FNAME                      EMP_LNAME
---------- ------------------------------ ------------------------------
JOB_TITLE                      JOB_SALARY
------------------------------ ----------
         1 JINCHEN                        YAO
GENERAL MANAGER                     70000
```

/* Sub-query 3:   Show job that has highest salary in the company. */

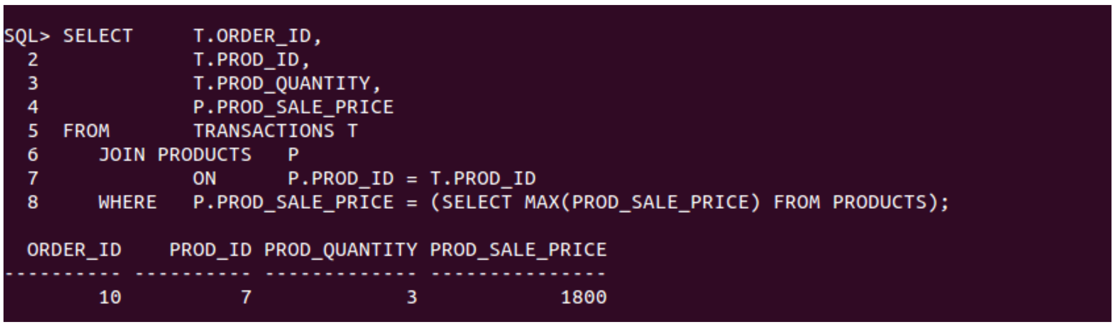
```
SELECT      JOB_TITLE,
            JOB_SALARY
FROM    JOBS
      WHERE      JOB_SALARY = (SELECT MAX(JOB_SALARY) FROM JOBS);
```



```
SQL> SELECT     JOB_TITLE,
  2             JOB_SALARY
  3  FROM       JOBS
  4     WHERE   JOB_SALARY = (SELECT MAX(JOB_SALARY) FROM JOBS);

JOB_TITLE                      JOB_SALARY
------------------------------ ----------
QUANT                               90000
```

/* Sub-query 4: Show the max sale price of all products. */

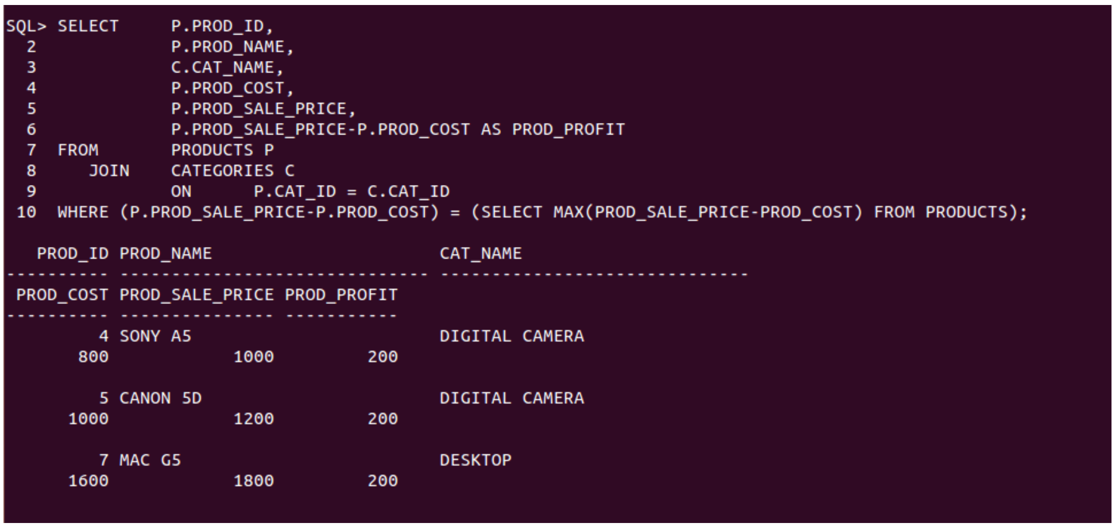
```
SELECT      T.ORDER_ID,
            T.PROD_ID,
            T.PROD_QUANTITY,
            P.PROD_SALE_PRICE
FROM    TRANSACTIONS T
      JOIN PRODUCTS     P
            ON        P.PROD_ID = T.PROD_ID
      WHERE      P.PROD_SALE_PRICE  =  (SELECT  MAX(PROD_SALE_PRICE)
FROM PRODUCTS);
```



```
SQL> SELECT     T.ORDER_ID,
                T.PROD_ID,
                T.PROD_QUANTITY,
                P.PROD_SALE_PRICE
     FROM       TRANSACTIONS T
        JOIN PRODUCTS   P
                ON      P.PROD_ID = T.PROD_ID
        WHERE   P.PROD_SALE_PRICE = (SELECT MAX(PROD_SALE_PRICE) FROM PRODUCTS);

  ORDER_ID    PROD_ID PROD_QUANTITY PROD_SALE_PRICE
---------- ---------- ------------- ---------------
        10          7             3            1800
```

/* Sub-query 5: Find the product that mskes the most money for the shop.   */

```
SELECT      P.PROD_ID,
            P.PROD_NAME,
            C.CAT_NAME,
            P.PROD_COST,
            P.PROD_SALE_PRICE,
            P.PROD_SALE_PRICE-P.PROD_COST AS PROD_PROFIT
FROM        PRODUCTS P
      JOIN  CATEGORIES C
            ON P.CAT_ID = C.CAT_ID
WHERE (P.PROD_SALE_PRICE-P.PROD_COST) = (SELECT MAX(PROD_SALE_PRICE-PROD_COST) FROM PRODUCTS);
```



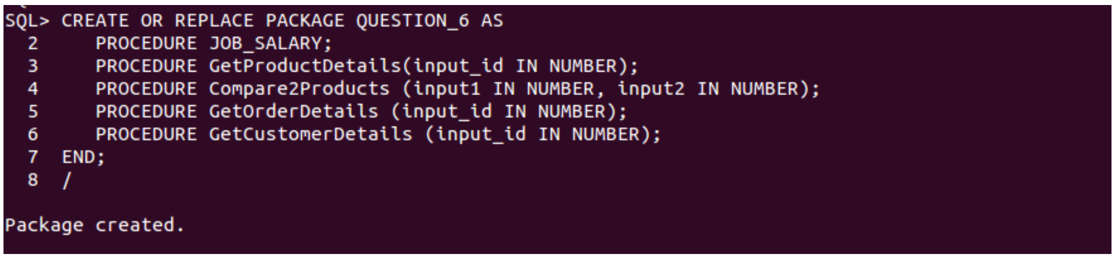
```
SQL> SELECT     P.PROD_ID,
                P.PROD_NAME,
                C.CAT_NAME,
                P.PROD_COST,
                P.PROD_SALE_PRICE,
                P.PROD_SALE_PRICE-P.PROD_COST AS PROD_PROFIT
     FROM       PRODUCTS P
        JOIN    CATEGORIES C
                ON      P.CAT_ID = C.CAT_ID
     WHERE (P.PROD_SALE_PRICE-P.PROD_COST) = (SELECT MAX(PROD_SALE_PRICE-PROD_COST) FROM PRODUCTS);

   PROD_ID PROD_NAME                      CAT_NAME
---------- ------------------------------ ------------------------------
 PROD_COST PROD_SALE_PRICE PROD_PROFIT
---------- --------------- -----------
         4 SONY A5                        DIGITAL CAMERA
       800            1000         200

         5 CANON 5D                       DIGITAL CAMERA
      1000            1200         200

         7 MAC G5                         DESKTOP
      1600            1800         200
```

6.   5 PL/SQL procedures as part of one package. One procedure must demonstrate each of the following:

- **The use of a cursor**
- **The use of save points**
- **The use of a rollback**

--CREATE THE PACKAGE

```
CREATE OR REPLACE PACKAGE QUESTION_6 AS
    PROCEDURE JOB_SALARY;
    PROCEDURE GetProductDetails(input_id IN NUMBER);
    PROCEDURE Compare2Products (input1 IN NUMBER, input2 IN NUMBER);
    PROCEDURE GetOrderDetails (input_id IN NUMBER);
    PROCEDURE GetCustomerDetails (input_id IN NUMBER);
END;
/
```



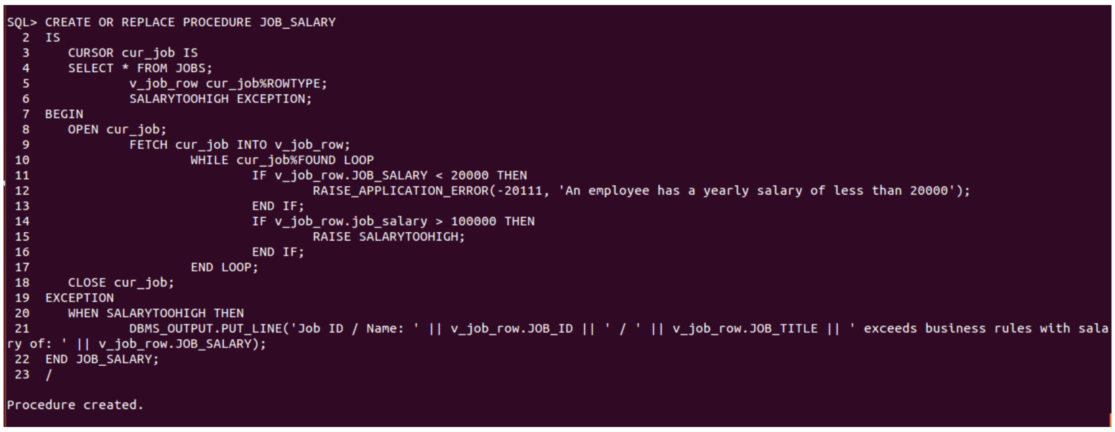
```
SQL> CREATE OR REPLACE PACKAGE QUESTION_6 AS
    PROCEDURE JOB_SALARY;
    PROCEDURE GetProductDetails(input_id IN NUMBER);
    PROCEDURE Compare2Products (input1 IN NUMBER, input2 IN NUMBER);
    PROCEDURE GetOrderDetails (input_id IN NUMBER);
    PROCEDURE GetCustomerDetails (input_id IN NUMBER);
END;
/

Package created.
```

--CREATE THE BODY OF THE PACKAGE

```
CREATE OR REPLACE PACKAGE BODY QUESTION_6 AS
```

/* PL/SQL Procedure 1 (with cursor): If any job is assigned a salary of less than 20,000, an application error will be raised while if any employee has a salary greater than 100,000, an exception will be raised. */

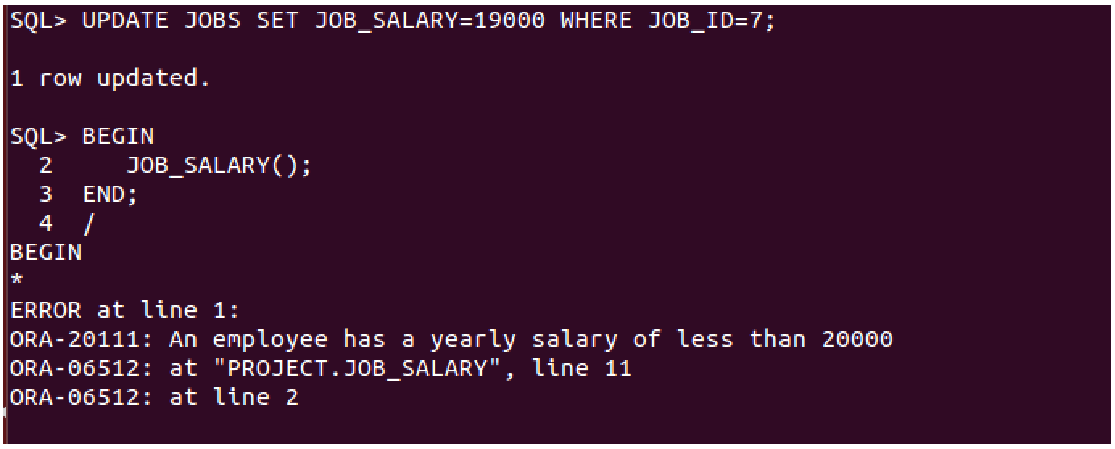
```
PROCEDURE JOB_SALARY
IS
    /* declare cursor, %rowtype and exception */
    CURSOR cur_job IS
        SELECT * FROM JOBS;
        v_job_row cur_job%ROWTYPE;
        SALARYTOOHIGH EXCEPTION;

BEGIN
    OPEN cur_job;
        FETCH cur_job INTO v_job_row;
            WHILE cur_job%FOUND LOOP
                IF v_job_row.JOB_SALARY < 20000 THEN
                    /* raise error if < 20000 */
                    RAISE_APPLICATION_ERROR(-20111, 'An employee has a
yearly salary of less than 20000');
                END IF;
                IF v_job_row.job_salary > 100000 THEN
                    /* raise exception if > 100000 */
                    RAISE SALARYTOOHIGH;
                END IF;
                FETCH cur_job INTO v_job_row;
            END LOOP;
    CLOSE cur_job;
/* define exception */
EXCEPTION
    WHEN SALARYTOOHIGH THEN
        DBMS_OUTPUT.PUT_LINE('Job ID / Name: ' || v_job_row.JOB_ID || ' / ' ||
v_job_row.JOB_TITLE || ' exceeds business rules with salary of: ' ||
v_job_row.JOB_SALARY);
END JOB_SALARY;
/
```



```
SQL> CREATE OR REPLACE PROCEDURE JOB_SALARY
  2  IS
  3     CURSOR cur_job IS
  4     SELECT * FROM JOBS;
  5             v_job_row cur_job%ROWTYPE;
  6             SALARYTOOHIGH EXCEPTION;
  7  BEGIN
  8     OPEN cur_job;
  9             FETCH cur_job INTO v_job_row;
 10                     WHILE cur_job%FOUND LOOP
 11                             IF v_job_row.JOB_SALARY < 20000 THEN
 12                                     RAISE_APPLICATION_ERROR(-20111, 'An employee has a yearly salary of less than 20000');
 13                             END IF;
 14                             IF v_job_row.job_salary > 100000 THEN
 15                                     RAISE SALARYTOOHIGH;
 16                             END IF;
 17                     END LOOP;
 18     CLOSE cur_job;
 19  EXCEPTION
 20     WHEN SALARYTOOHIGH THEN
 21             DBMS_OUTPUT.PUT_LINE('Job ID / Name: ' || v_job_row.JOB_ID || ' / ' || v_job_row.JOB_TITLE || ' exceeds business rules with sala
ry of: ' || v_job_row.JOB_SALARY);
 22  END JOB_SALARY;
 23  /

Procedure created.
```

/* update employee salary to 19000 */

```
UPDATE JOBS SET JOB_SALARY=19000 WHERE JOB_ID=7;
BEGIN
    JOB_SALARY();
END;
/
```

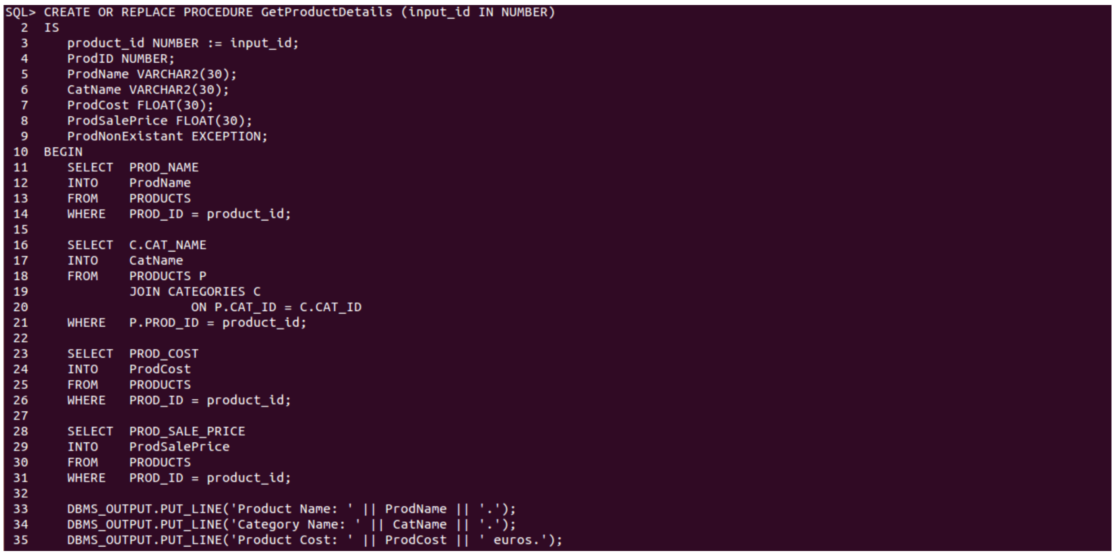


```
SQL> UPDATE JOBS SET JOB_SALARY=19000 WHERE JOB_ID=7;

1 row updated.

SQL> BEGIN
  2     JOB_SALARY();
  3  END;
  4  /
BEGIN
*
ERROR at line 1:
ORA-20111: An employee has a yearly salary of less than 20000
ORA-06512: at "PROJECT.JOB_SALARY", line 11
ORA-06512: at line 2
```

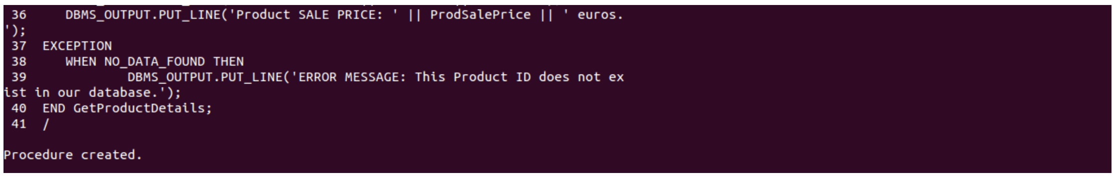
/* PL/SQL Procedure 2: Display a products details when its ID is entered in as an input parameter. Output the name, description, category and cost of the product. */

```
CREATE OR REPLACE PROCEDURE GetProductDetails (input_id IN NUMBER)
IS
	product_id NUMBER := input_id;
	ProdID NUMBER;
	ProdName VARCHAR2(30);
	CatName VARCHAR2(30);
	ProdCost FLOAT(30);
	ProdSalePrice FLOAT(30);
	ProdNonExistant EXCEPTION;

BEGIN
	SELECT	PROD_NAME
	INTO	ProdName
	FROM	PRODUCTS
	WHERE	PROD_ID = product_id;

	SELECT	C.CAT_NAME
	INTO	CatName
	FROM	PRODUCTS P
		JOIN CATEGORIES C
			ON P.CAT_ID = C.CAT_ID
	WHERE	P.PROD_ID = product_id;

	SELECT	PROD_COST
	INTO	ProdCost
	FROM	PRODUCTS
	WHERE	PROD_ID = product_id;

	SELECT	PROD_SALE_PRICE
	INTO	ProdSalePrice
	FROM	PRODUCTS
	WHERE	PROD_ID = product_id;

	DBMS_OUTPUT.PUT_LINE('Product Name: ' || ProdName || '.');
	DBMS_OUTPUT.PUT_LINE('Category Name: ' || CatName || '.');
	DBMS_OUTPUT.PUT_LINE('Product Cost: ' || ProdCost || ' euros.');
       DBMS_OUTPUT.PUT_LINE('Product SALE PRICE: ' || ProdSalePrice || ' euros.');

EXCEPTION
	WHEN NO_DATA_FOUND THEN
```

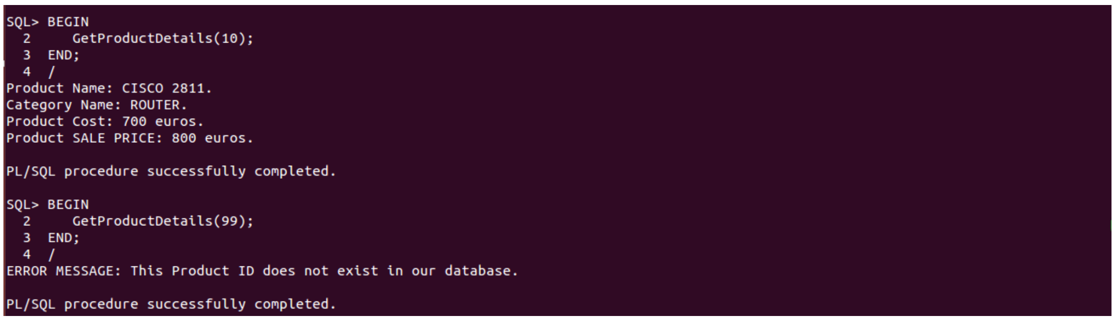
```
        DBMS_OUTPUT.PUT_LINE('ERROR MESSAGE: This Product ID does not exist in our database.');

END GetProductDetails;
/

BEGIN
    GetProductDetails(10);
END;
/

BEGIN
    GetProductDetails(99);
END;
/
```



```
SQL> CREATE OR REPLACE PROCEDURE GetProductDetails (input_id IN NUMBER)
     IS
        product_id NUMBER := input_id;
        ProdID NUMBER;
        ProdName VARCHAR2(30);
        CatName VARCHAR2(30);
        ProdCost FLOAT(30);
        ProdSalePrice FLOAT(30);
        ProdNonExistant EXCEPTION;
     BEGIN
        SELECT  PROD_NAME
        INTO    ProdName
        FROM    PRODUCTS
        WHERE   PROD_ID = product_id;

        SELECT  C.CAT_NAME
        INTO    CatName
        FROM    PRODUCTS P
                JOIN CATEGORIES C
                        ON P.CAT_ID = C.CAT_ID
        WHERE   P.PROD_ID = product_id;

        SELECT  PROD_COST
        INTO    ProdCost
        FROM    PRODUCTS
        WHERE   PROD_ID = product_id;

        SELECT  PROD_SALE_PRICE
        INTO    ProdSalePrice
        FROM    PRODUCTS
        WHERE   PROD_ID = product_id;

        DBMS_OUTPUT.PUT_LINE('Product Name: ' || ProdName || '.');
        DBMS_OUTPUT.PUT_LINE('Category Name: ' || CatName || '.');
        DBMS_OUTPUT.PUT_LINE('Product Cost: ' || ProdCost || ' euros.');
```



```
        DBMS_OUTPUT.PUT_LINE('Product SALE PRICE: ' || ProdSalePrice || ' euros.
');
     EXCEPTION
        WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('ERROR MESSAGE: This Product ID does not ex
ist in our database.');
     END GetProductDetails;
     /

Procedure created.
```



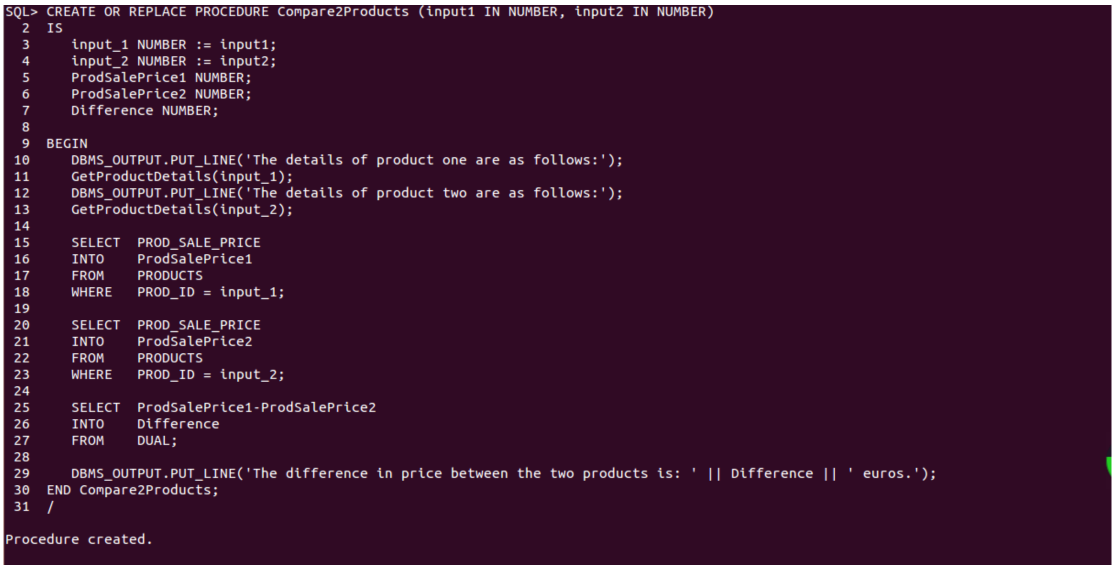
```
SQL> BEGIN
        GetProductDetails(10);
     END;
     /
Product Name: CISCO 2811.
Category Name: ROUTER.
Product Cost: 700 euros.
Product SALE PRICE: 800 euros.

PL/SQL procedure successfully completed.

SQL> BEGIN
        GetProductDetails(99);
     END;
     /
ERROR MESSAGE: This Product ID does not exist in our database.

PL/SQL procedure successfully completed.
```

/* PL/SQL Procedure 3: Compare 2 products side by side and show price

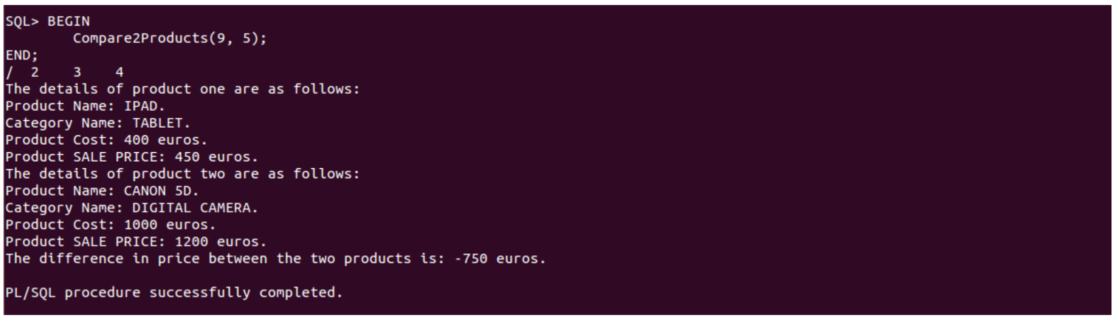
difference. Use previously created procedure within this procedure. */

```
CREATE OR REPLACE PROCEDURE Compare2Products (input1 IN NUMBER,
input2 IN NUMBER)
IS

    input_1 NUMBER := input1;
    input_2 NUMBER := input2;
    ProdCost1 NUMBER;
    ProdCost2 NUMBER;
    Answer NUMBER;

BEGIN

    DBMS_OUTPUT.PUT_LINE('The details of product one are as follows:');
    GetProductDetails(input_1);
    DBMS_OUTPUT.PUT_LINE('The details of product two are as follows:');
    GetProductDetails(input_2);

    SELECT    prod_sale_price
    INTO   ProdCost1
    FROM   products
    WHERE     prod_id = input_1;

    SELECT    prod_sale_price
    INTO   ProdCost2
    FROM   products
    WHERE     prod_id = input_2;

    SELECT    ProdCost1-ProdCost2
    INTO   Answer
    FROM   dual;

    /* note that Answer may contain a minus if product two is more than product one
*/
    DBMS_OUTPUT.PUT_LINE('The difference in price between the two products
is: ' || Answer || ' euros.');
    /* exception handling is provided in the above procedures */

END Compare2Products;
/

BEGIN
    Compare2Products(63, 62);
```

```
END;
/
```



```
SQL> CREATE OR REPLACE PROCEDURE Compare2Products (input1 IN NUMBER, input2 IN NUMBER)
     IS
        input_1 NUMBER := input1;
        input_2 NUMBER := input2;
        ProdSalePrice1 NUMBER;
        ProdSalePrice2 NUMBER;
        Difference NUMBER;

     BEGIN
        DBMS_OUTPUT.PUT_LINE('The details of product one are as follows:');
        GetProductDetails(input_1);
        DBMS_OUTPUT.PUT_LINE('The details of product two are as follows:');
        GetProductDetails(input_2);

        SELECT  PROD_SALE_PRICE
        INTO    ProdSalePrice1
        FROM    PRODUCTS
        WHERE   PROD_ID = input_1;

        SELECT  PROD_SALE_PRICE
        INTO    ProdSalePrice2
        FROM    PRODUCTS
        WHERE   PROD_ID = input_2;

        SELECT  ProdSalePrice1-ProdSalePrice2
        INTO    Difference
        FROM    DUAL;

        DBMS_OUTPUT.PUT_LINE('The difference in price between the two products is: ' || Difference || ' euros.');
     END Compare2Products;
     /

Procedure created.
```



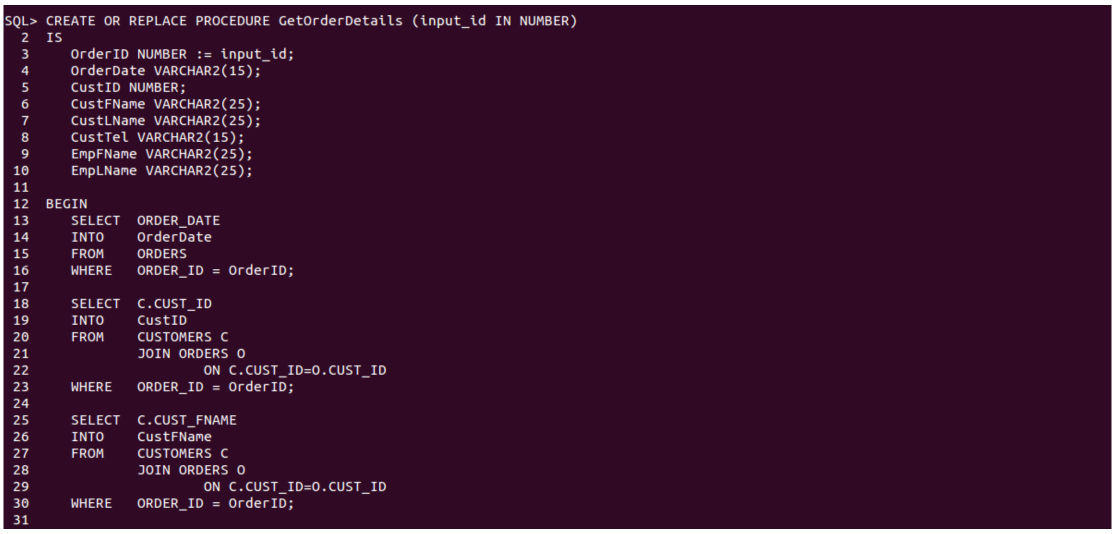
```
SQL> BEGIN
        Compare2Products(9, 5);
END;
/  2    3    4
The details of product one are as follows:
Product Name: IPAD.
Category Name: TABLET.
Product Cost: 400 euros.
Product SALE PRICE: 450 euros.
The details of product two are as follows:
Product Name: CANON 5D.
Category Name: DIGITAL CAMERA.
Product Cost: 1000 euros.
Product SALE PRICE: 1200 euros.
The difference in price between the two products is: -750 euros.

PL/SQL procedure successfully completed.
```

/* PL/SQL Procedure 4: Show full order details when order ID is entered. */

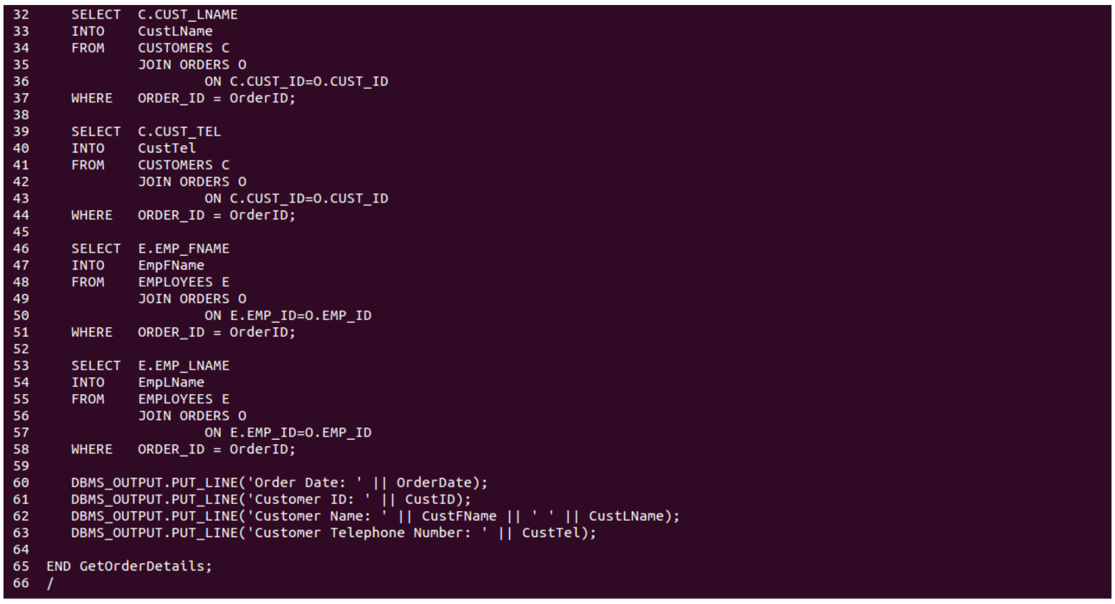
```
CREATE OR REPLACE PROCEDURE GetOrderDetails (input_id IN NUMBER)
IS
    OrderID NUMBER := input_id;
    OrderDate VARCHAR2(15);
    CustID NUMBER;
    CustFName VARCHAR2(25);
    CustLName VARCHAR2(25);
    CustTel VARCHAR2(15);
    EmpFName VARCHAR2(25);
    EmpLName VARCHAR2(25);

BEGIN
    SELECT    ORDER_DATE
    INTO      OrderDate
    FROM      ORDERS
```

```
WHERE      ORDER_ID = OrderID;

SELECT     C.CUST_ID
INTO    CustID
FROM   CUSTOMERS C
    JOIN ORDERS O
        ON C.CUST_ID=O.CUST_ID
WHERE      ORDER_ID = OrderID;

SELECT     C.CUST_FNAME
INTO    CustFName
FROM   CUSTOMERS C
    JOIN ORDERS O
        ON C.CUST_ID=O.CUST_ID
WHERE      ORDER_ID = OrderID;

SELECT     C.CUST_LNAME
INTO    CustLName
FROM   CUSTOMERS C
    JOIN ORDERS O
        ON C.CUST_ID=O.CUST_ID
WHERE      ORDER_ID = OrderID;

SELECT     C.CUST_TEL
INTO    CustTel
FROM   CUSTOMERS C
    JOIN ORDERS O
        ON C.CUST_ID=O.CUST_ID
WHERE      ORDER_ID = OrderID;

SELECT     E.EMP_FNAME
INTO    EmpFName
FROM   EMPLOYEES E
    JOIN ORDERS O
        ON E.EMP_ID=O.EMP_ID
WHERE      ORDER_ID = OrderID;

SELECT     E.EMP_LNAME
INTO    EmpLName
FROM   EMPLOYEES E
    JOIN ORDERS O
        ON E.EMP_ID=O.EMP_ID
WHERE      ORDER_ID = OrderID;
```

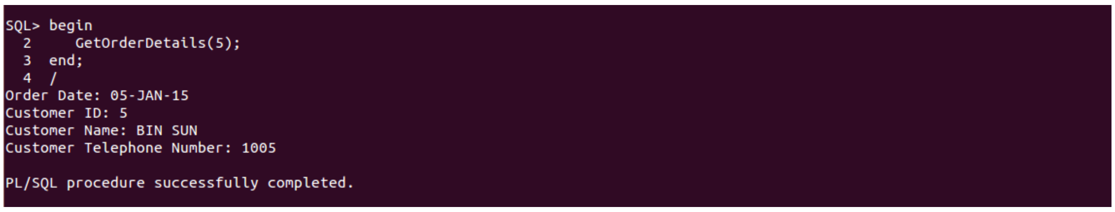
```
    DBMS_OUTPUT.PUT_LINE('Order Date: ' || OrderDate);
    DBMS_OUTPUT.PUT_LINE('Customer ID: ' || CustID);
    DBMS_OUTPUT.PUT_LINE('Customer Name: ' || CustFName || ' ' || CustLName);
    DBMS_OUTPUT.PUT_LINE('Customer Telephone Number: ' || CustTel);
    DBMS_OUTPUT.PUT_LINE('Employee In Charge: ' || EmpFName || ' ' || EmpLName);

END GetOrderDetails;
/

BEGIN
    GetOrderDetails(5);
END;
/
```



```
SQL> CREATE OR REPLACE PROCEDURE GetOrderDetails (input_id IN NUMBER)
IS
   OrderID NUMBER := input_id;
   OrderDate VARCHAR2(15);
   CustID NUMBER;
   CustFName VARCHAR2(25);
   CustLName VARCHAR2(25);
   CustTel VARCHAR2(15);
   EmpFName VARCHAR2(25);
   EmpLName VARCHAR2(25);

BEGIN
   SELECT  ORDER_DATE
   INTO    OrderDate
   FROM    ORDERS
   WHERE   ORDER_ID = OrderID;

   SELECT  C.CUST_ID
   INTO    CustID
   FROM    CUSTOMERS C
           JOIN ORDERS O
                   ON C.CUST_ID=O.CUST_ID
   WHERE   ORDER_ID = OrderID;

   SELECT  C.CUST_FNAME
   INTO    CustFName
   FROM    CUSTOMERS C
           JOIN ORDERS O
                   ON C.CUST_ID=O.CUST_ID
   WHERE   ORDER_ID = OrderID;

```



```
   SELECT  C.CUST_LNAME
   INTO    CustLName
   FROM    CUSTOMERS C
           JOIN ORDERS O
                   ON C.CUST_ID=O.CUST_ID
   WHERE   ORDER_ID = OrderID;

   SELECT  C.CUST_TEL
   INTO    CustTel
   FROM    CUSTOMERS C
           JOIN ORDERS O
                   ON C.CUST_ID=O.CUST_ID
   WHERE   ORDER_ID = OrderID;

   SELECT  E.EMP_FNAME
   INTO    EmpFName
   FROM    EMPLOYEES E
           JOIN ORDERS O
                   ON E.EMP_ID=O.EMP_ID
   WHERE   ORDER_ID = OrderID;

   SELECT  E.EMP_LNAME
   INTO    EmpLName
   FROM    EMPLOYEES E
           JOIN ORDERS O
                   ON E.EMP_ID=O.EMP_ID
   WHERE   ORDER_ID = OrderID;

   DBMS_OUTPUT.PUT_LINE('Order Date: ' || OrderDate);
   DBMS_OUTPUT.PUT_LINE('Customer ID: ' || CustID);
   DBMS_OUTPUT.PUT_LINE('Customer Name: ' || CustFName || ' ' || CustLName);
   DBMS_OUTPUT.PUT_LINE('Customer Telephone Number: ' || CustTel);

END GetOrderDetails;
/
```

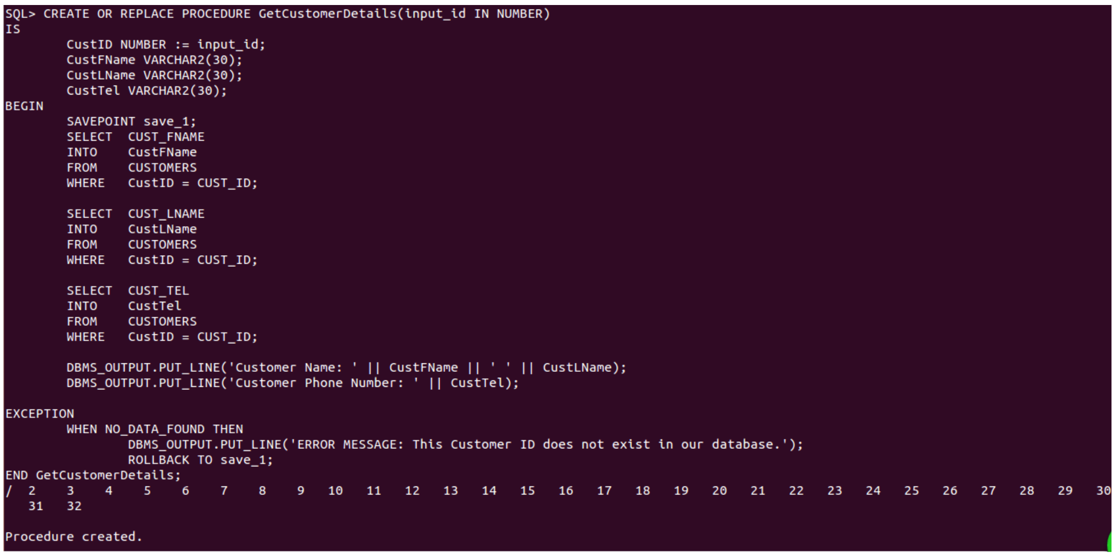


```
SQL> begin
  2     GetOrderDetails(5);
  3  end;
  4  /
Order Date: 05-JAN-15
Customer ID: 5
Customer Name: BIN SUN
Customer Telephone Number: 1005

PL/SQL procedure successfully completed.
```

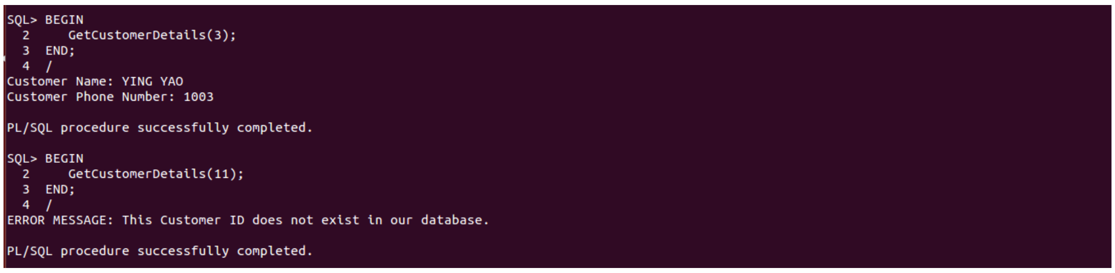
/* PL/SQL Procedure 5: Get customer details when a Customer's ID is entered. It will display any orders ID's associated with that customer.(with savepoint and rollback) */

```
CREATE OR REPLACE PROCEDURE GetCustomerDetails(input_id IN NUMBER)
IS
    CustID NUMBER := input_id;
    CustFName VARCHAR2(30);
    CustLName VARCHAR2(30);
    CustTel VARCHAR2(30);
BEGIN
    SAVEPOINT save_1;
    SELECT      CUST_FNAME
    INTO    CustFName
    FROM    CUSTOMERS
    WHERE       CustID = CUST_ID;

    SELECT      CUST_LNAME
    INTO    CustLName
    FROM    CUSTOMERS
    WHERE       CustID = CUST_ID;

    SELECT      CUST_TEL
    INTO    CustTel
    FROM    CUSTOMERS
    WHERE       CustID = CUST_ID;

    DBMS_OUTPUT.PUT_LINE('Customer Name: ' || CustFName || ' ' || CustLName);
    DBMS_OUTPUT.PUT_LINE('Customer Phone Number: ' || CustTel);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('ERROR MESSAGE: This Customer ID does not exist in our database.');
        ROLLBACK TO save_1;
END GetCustomerDetails;
/
```

```
BEGIN
	GetCustomerDetails(3);
END;
/

BEGIN
	GetCustomerDetails(11);
END;
/
```



```
SQL> CREATE OR REPLACE PROCEDURE GetCustomerDetails(input_id IN NUMBER)
IS
        CustID NUMBER := input_id;
        CustFName VARCHAR2(30);
        CustLName VARCHAR2(30);
        CustTel VARCHAR2(30);
BEGIN
        SAVEPOINT save_1;
        SELECT  CUST_FNAME
        INTO    CustFName
        FROM    CUSTOMERS
        WHERE   CustID = CUST_ID;

        SELECT  CUST_LNAME
        INTO    CustLName
        FROM    CUSTOMERS
        WHERE   CustID = CUST_ID;

        SELECT  CUST_TEL
        INTO    CustTel
        FROM    CUSTOMERS
        WHERE   CustID = CUST_ID;

        DBMS_OUTPUT.PUT_LINE('Customer Name: ' || CustFName || ' ' || CustLName);
        DBMS_OUTPUT.PUT_LINE('Customer Phone Number: ' || CustTel);

EXCEPTION
        WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('ERROR MESSAGE: This Customer ID does not exist in our database.');
                ROLLBACK TO save_1;
END GetCustomerDetails;
/

Procedure created.
```



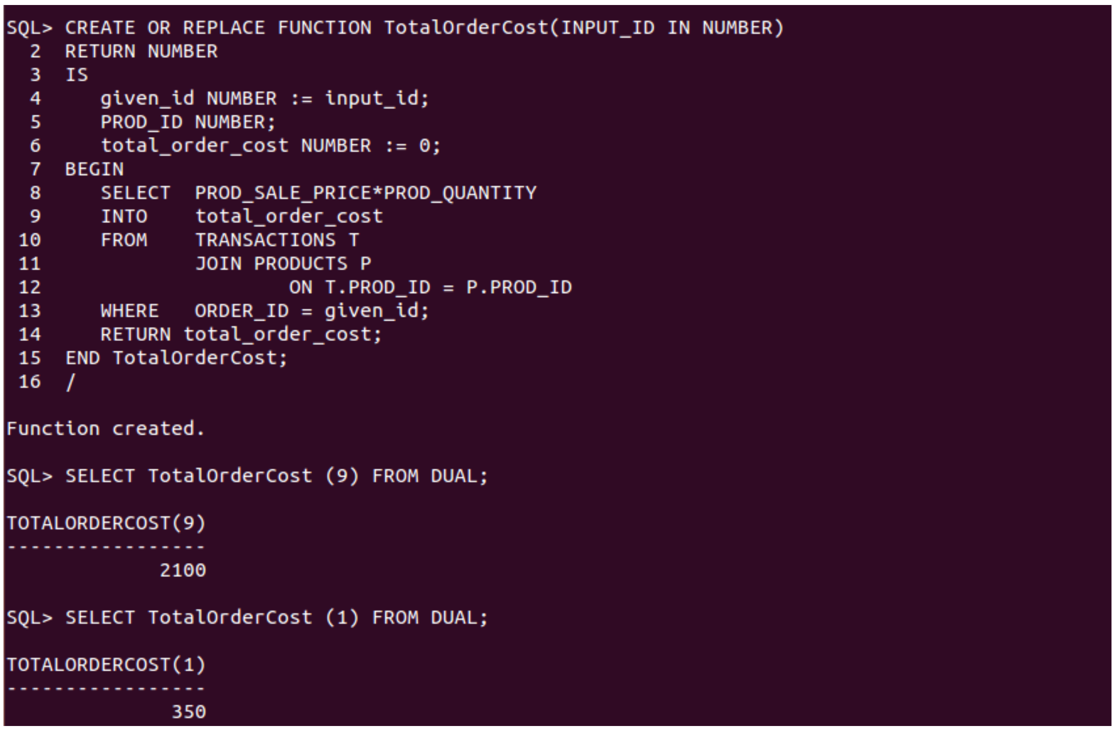
```
SQL> BEGIN
       GetCustomerDetails(3);
     END;
     /
Customer Name: YING YAO
Customer Phone Number: 1003

PL/SQL procedure successfully completed.

SQL> BEGIN
       GetCustomerDetails(11);
     END;
     /
ERROR MESSAGE: This Customer ID does not exist in our database.

PL/SQL procedure successfully completed.
```

**/*	2 PL/SQL function	*/**

/* 1 PL/SQL function: Allow user to input Order ID, then output the total order cost of that product. */

```
CREATE OR REPLACE FUNCTION TotalOrderCost(input_id IN NUMBER)
RETURN NUMBER
IS
	given_id NUMBER := input_id;
	PROD_ID NUMBER;
	total_order_cost NUMBER := 0;
BEGIN
```

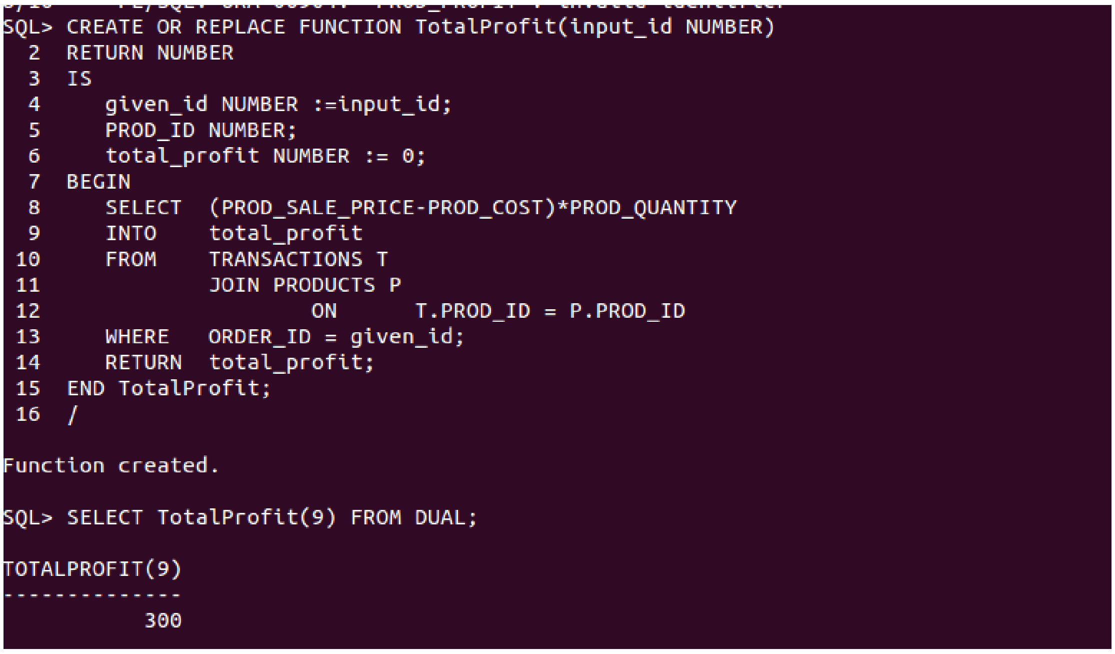
```
    /* select sum of order */
    SELECT      PROD_SALE_PRICE*PROD_QUANTITY
    INTO     total_order_cost
    FROM   TRANSACTIONS T
        JOIN PRODUCTS P
            ON T.PROD_ID = P.PROD_ID
    WHERE       ORDER_ID = given_id;

    RETURN total_order_cost;
END TotalOrderCost;
/

SELECT TotalOrderCost(9) FROM DUAL;
SELECT TotalOrderCost(1) FROM DUAL;
```



```
SQL> CREATE OR REPLACE FUNCTION TotalOrderCost(INPUT_ID IN NUMBER)
  RETURN NUMBER
  IS
     given_id NUMBER := input_id;
     PROD_ID NUMBER;
     total_order_cost NUMBER := 0;
  BEGIN
     SELECT  PROD_SALE_PRICE*PROD_QUANTITY
     INTO    total_order_cost
     FROM    TRANSACTIONS T
             JOIN PRODUCTS P
                     ON T.PROD_ID = P.PROD_ID
     WHERE   ORDER_ID = given_id;
     RETURN total_order_cost;
  END TotalOrderCost;
  /

Function created.

SQL> SELECT TotalOrderCost (9) FROM DUAL;

TOTALORDERCOST(9)
-----------------
             2100

SQL> SELECT TotalOrderCost (1) FROM DUAL;

TOTALORDERCOST(1)
-----------------
              350
```

/* 2 PL/SQL function: Allow user to input Order ID, then output the total profit of that product. */

```
CREATE OR REPLACE FUNCTION TotalProfit(input_id NUMBER)
RETURN NUMBER
IS
    given_id NUMBER :=input_id;
    PROD_ID NUMBER;
    total_profit NUMBER := 0;
BEGIN
    SELECT      (PROD_SALE_PRICE-PROD_COST)*PROD_QUANTITY
    INTO     total_profit
    FROM   TRANSACTIONS T
        JOIN PRODUCTS P
            ON        T.PROD_ID = P.PROD_ID
    WHERE       ORDER_ID = given_id;

    RETURN     total_profit;
END TotalProfit;
/

SELECT TotalProfit(9) FROM DUAL;
```



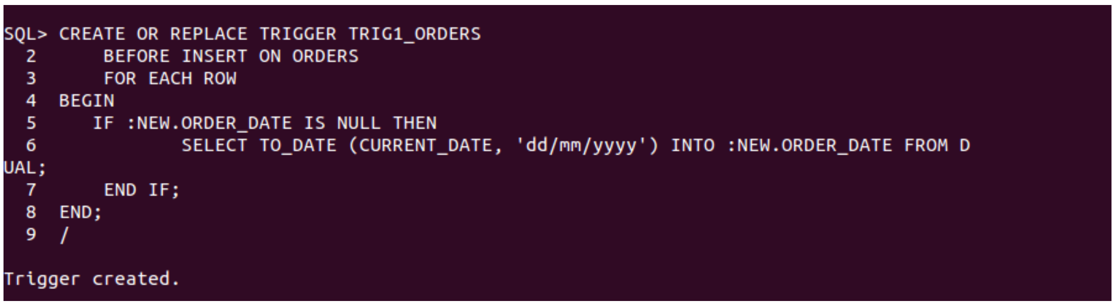
```
SQL> CREATE OR REPLACE FUNCTION TotalProfit(input_id NUMBER)
     RETURN NUMBER
     IS
        given_id NUMBER :=input_id;
        PROD_ID NUMBER;
        total_profit NUMBER := 0;
     BEGIN
        SELECT  (PROD_SALE_PRICE-PROD_COST)*PROD_QUANTITY
        INTO    total_profit
        FROM    TRANSACTIONS T
                JOIN PRODUCTS P
                        ON      T.PROD_ID = P.PROD_ID
        WHERE   ORDER_ID = given_id;
        RETURN  total_profit;
     END TotalProfit;
     /

Function created.

SQL> SELECT TotalProfit(9) FROM DUAL;

TOTALPROFIT(9)
--------------
           300
```

/* 3 Triggers (at least 1 before, and at least 1 after) */

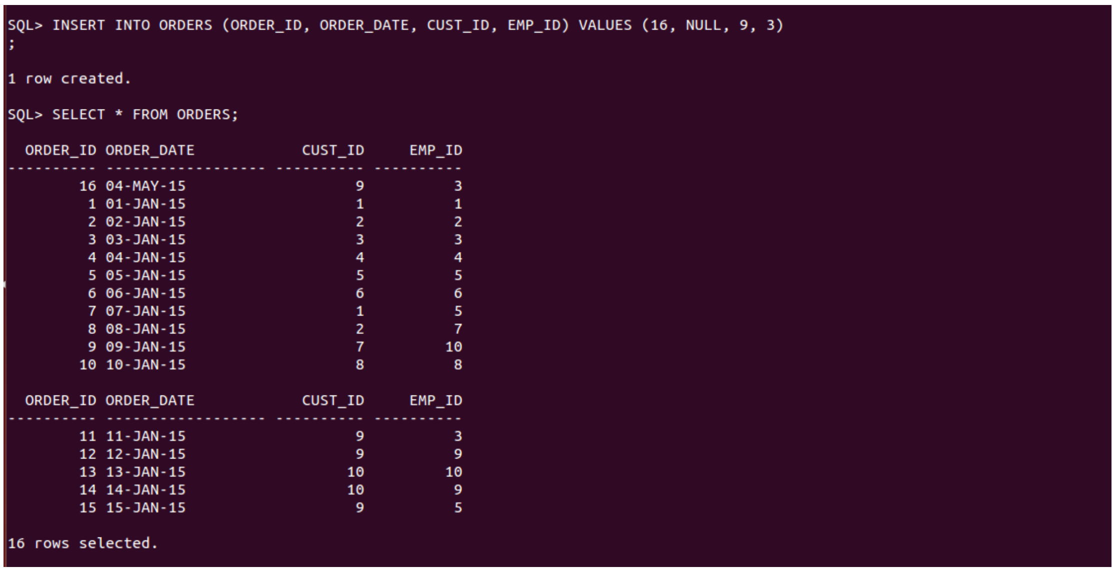
/* Trigger 1 (before): If no order date has been entered into the order table, automatically gendererate this based on the current date and insert that. */

```
CREATE OR REPLACE TRIGGER TRIG1_ORDERS
     BEFORE INSERT ON ORDERS
     FOR EACH ROW
BEGIN
    IF :NEW.ORDER_DATE = NULL THEN
        SELECT     TO_DATE     (CURRENT_DATE,     'dd/mm/yyyy')
INTO :NEW.ORDER_DATE FROM DUAL;
     END IF;
END;
```

/
INSERT INTO ORDERS (ORDER_ID, ORDER_DATE, CUST_ID, EMP_ID) VALUES (16, NULL, 9, 3);
SELECT * FROM ORDERS;



```
SQL> CREATE OR REPLACE TRIGGER TRIG1_ORDERS
       BEFORE INSERT ON ORDERS
       FOR EACH ROW
     BEGIN
       IF :NEW.ORDER_DATE IS NULL THEN
             SELECT TO_DATE (CURRENT_DATE, 'dd/mm/yyyy') INTO :NEW.ORDER_DATE FROM D
UAL;
       END IF;
     END;
     /

Trigger created.
```



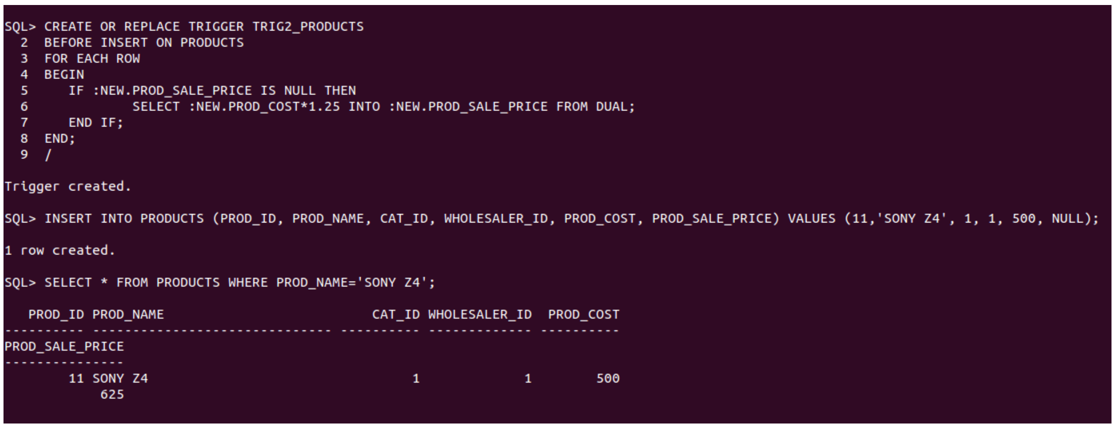
```
SQL> INSERT INTO ORDERS (ORDER_ID, ORDER_DATE, CUST_ID, EMP_ID) VALUES (16, NULL, 9, 3)
;

1 row created.

SQL> SELECT * FROM ORDERS;

  ORDER_ID ORDER_DATE           CUST_ID     EMP_ID
---------- ------------------ ---------- ----------
        16 04-MAY-15                   9          3
         1 01-JAN-15                   1          1
         2 02-JAN-15                   2          2
         3 03-JAN-15                   3          3
         4 04-JAN-15                   4          4
         5 05-JAN-15                   5          5
         6 06-JAN-15                   6          6
         7 07-JAN-15                   1          5
         8 08-JAN-15                   2          7
         9 09-JAN-15                   7         10
        10 10-JAN-15                   8          8

  ORDER_ID ORDER_DATE           CUST_ID     EMP_ID
---------- ------------------ ---------- ----------
        11 11-JAN-15                   9          3
        12 12-JAN-15                   9          9
        13 13-JAN-15                  10         10
        14 14-JAN-15                  10          9
        15 15-JAN-15                   9          5

16 rows selected.
```

/* Trigger 2 (before): If no sale price has been entered into the database, generate it by multiplying the wholesale price by 1.25. */

CREATE OR REPLACE TRIGGER TRIG2_PRODUCTS
BEFORE INSERT ON PRODUCTS
FOR EACH ROW
BEGIN
    IF :NEW.PROD_SALE_PRICE IS NULL THEN
        SELECT :NEW.PROD_COST*1.25 INTO :NEW.PROD_SALE_PRICE FROM DUAL;
    END IF;
END;
/

INSERT INTO PRODUCTS (PROD_ID, PROD_NAME, CAT_ID, WHOLESALER_ID, PROD_COST, PROD_SALE_PRICE) VALUES (11,'SONY Z4', 1, 1, 500, NULL);

SELECT * FROM PRODUCTS WHERE PROD_NAME='SONY Z4';

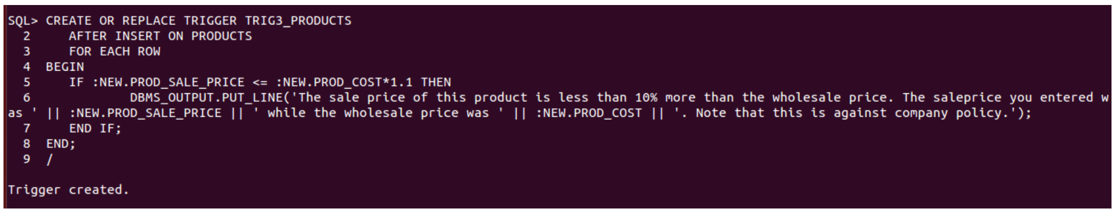


```
SQL> CREATE OR REPLACE TRIGGER TRIG2_PRODUCTS
     BEFORE INSERT ON PRODUCTS
     FOR EACH ROW
     BEGIN
        IF :NEW.PROD_SALE_PRICE IS NULL THEN
                SELECT :NEW.PROD_COST*1.25 INTO :NEW.PROD_SALE_PRICE FROM DUAL;
        END IF;
     END;
     /

Trigger created.

SQL> INSERT INTO PRODUCTS (PROD_ID, PROD_NAME, CAT_ID, WHOLESALER_ID, PROD_COST, PROD_SALE_PRICE) VALUES (11,'SONY Z4', 1, 1, 500, NULL);

1 row created.

SQL> SELECT * FROM PRODUCTS WHERE PROD_NAME='SONY Z4';

   PROD_ID PROD_NAME                          CAT_ID WHOLESALER_ID  PROD_COST
---------- ---------------------------- ---------- ------------- ----------
PROD_SALE_PRICE
---------------
        11 SONY Z4                                 1             1        500
            625
```

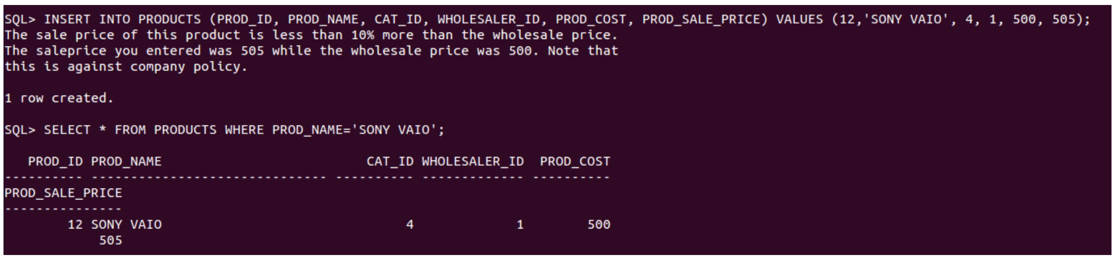
/* Trigger 3 (after): If a value is entered into the sale price column which is less than 10% more than what the wholesale price of that item is, then notift the user after it has been input into the database. */

```
CREATE OR REPLACE TRIGGER TRIG3_PRODUCTS
	AFTER INSERT ON PRODUCTS
	FOR EACH ROW
BEGIN
	IF :NEW.PROD_SALE_PRICE <= :NEW.PROD_COST*1.1 THEN
		DBMS_OUTPUT.PUT_LINE('The sale price of this product is less than 10% more than the wholesale price. The saleprice you entered was ' || :NEW.PROD_SALE_PRICE || ' while the wholesale price was ' || :NEW.PROD_COST || '. Note that this is against company policy.');
	END IF;
END;
/
```

INSERT INTO PRODUCTS (PROD_ID, PROD_NAME, CAT_ID, WHOLESALER_ID, PROD_COST, PROD_SALE_PRICE) VALUES (12,'SONY VAIO', 4, 1, 500, 505);
SELECT * FROM PRODUCTS WHERE PROD_NAME='SONY VAIO';



```
SQL> CREATE OR REPLACE TRIGGER TRIG3_PRODUCTS
        AFTER INSERT ON PRODUCTS
        FOR EACH ROW
     BEGIN
        IF :NEW.PROD_SALE_PRICE <= :NEW.PROD_COST*1.1 THEN
                DBMS_OUTPUT.PUT_LINE('The sale price of this product is less than 10% more than the wholesale price. The saleprice you entered w
as ' || :NEW.PROD_SALE_PRICE || ' while the wholesale price was ' || :NEW.PROD_COST || '. Note that this is against company policy.');
        END IF;
     END;
     /

Trigger created.
```



```
SQL> INSERT INTO PRODUCTS (PROD_ID, PROD_NAME, CAT_ID, WHOLESALER_ID, PROD_COST, PROD_SALE_PRICE) VALUES (12,'SONY VAIO', 4, 1, 500, 505);
The sale price of this product is less than 10% more than the wholesale price.
The saleprice you entered was 505 while the wholesale price was 500. Note that
this is against company policy.

1 row created.

SQL> SELECT * FROM PRODUCTS WHERE PROD_NAME='SONY VAIO';

   PROD_ID PROD_NAME                          CAT_ID WHOLESALER_ID  PROD_COST
---------- ------------------------------ ---------- ------------- ----------
PROD_SALE_PRICE
---------------
        12 SONY VAIO                               4             1        500
            505
```

/*9. Identification of weaknesses or potential improvements to the database */

There are a number of ways that the database could be developed in order to meet further company demands. For example, a postage system could be implemented that would automatically generate postage costs based on the weight of products being ordered and the address to which they are being shipped.

Also, stock levels could be saved in the database. When products are sold and when products are bought from wholesalers the database could automatically update stock levels to represent the updated stock levels.