Exam Data Submit Help 2 : 3 : 34

# DBMS Hands-on Assessment

**Duration: 2 hours**

**Max Marks: 20**

**Instructions to examinees:**

1. Trainee should not tamper/modify the default folder structure in eclipse. Any changes done to default folder structure in eclipse will result in solution scripts not being evaluated.
2. Trainee should take frequent backup of your solutions.
3. There is no "**Auto Submission**" feature and hence you(Trainee) have to submit the scripts mandatorily before the closing time of the exam.
4. Submission has to be done only once, after completing all the solutions in Eclipse.
5. Trainee have to write SQL solutions in the respective solution files provided for each question. Violating this would result in you getting Zero marks for the respective question.
6. Ensure that each file should contain only 1 Query.
7. Read the problem statement, functionality and the other details provided carefully and implement the solution.
8. Trainee need **NOT** create table(s) and insert records, as it has been already created and the necessary records have been populated in Eclipse plugin.
9. Once you have implemented the solution(s), save the solution(s) in Eclipse plugin.
10. Submit the assessment folder which will be there inside workspace of examid folder through hands on client.
11. The solutions provided by you must:
    a. be generic and independent of the given sample data.
    b. **NOT** be hard coded.
12. You can write DML statements (Insert, Update, Delete and Select) in test.sql that would be provided in SQL folder.
13. You can see the modified table data (if any done by you) by writing SELECT queries in test.sql that would be provided in SQL folder.
14. You can get original data by clicking on the Reload button.
15. Do **NOT** try to delete any of the files.
16. You must stop working when the invigilator asks you to do so.
17. Test.sql files are **NOT** evaluated.
18. Templates of solution files are available in your hands-on examination client. Download the same for writing solution in case Eclipse plugin didn't load. DO **NOT** create separate file for writing the solutions.
19. Column names are **case insensitive** in oracle and hence the column or alias names can be displayed in uppercase or lowercase.
20. The rows can be displayed in any order.

**A business scenario:**

"XYZ retail stores" has a vast collection of daily essential and beauty products. Customers buy various products by paying the amount through card or cash. The database programmers of the application for the retail store maintain the database to store the details about the customers, products and transactions of the retail store. The table structure and sample data for the tables in the database is given below.

The table **customer** provides the details about all the customers.

**Table: customer**

| Column name | Data type and size | Constraints | Description |
|---|---|---|---|

Exam Data Submit Help 2 : 3 : 3

| Column name | Data type and size | Constraints | Description |
|---|---|---|---|
| custid | NUMBER | PRIMARY KEY | unique id for customer |
| custname | VARCHAR2(30) | NOT NULL | name of the customer |
| city | VARCHAR2(30) | NOT NULL | city of the customer |
| cardid | VARCHAR2(5) | UNIQUE,CHECK | unique cardid, should start with 'C'. card is optional. |
| cardbalance | NUMBER | CHECK | balance amount in the card. cardbalance should be greater than equal to 0 |

The table **product** provides the details about all the products.

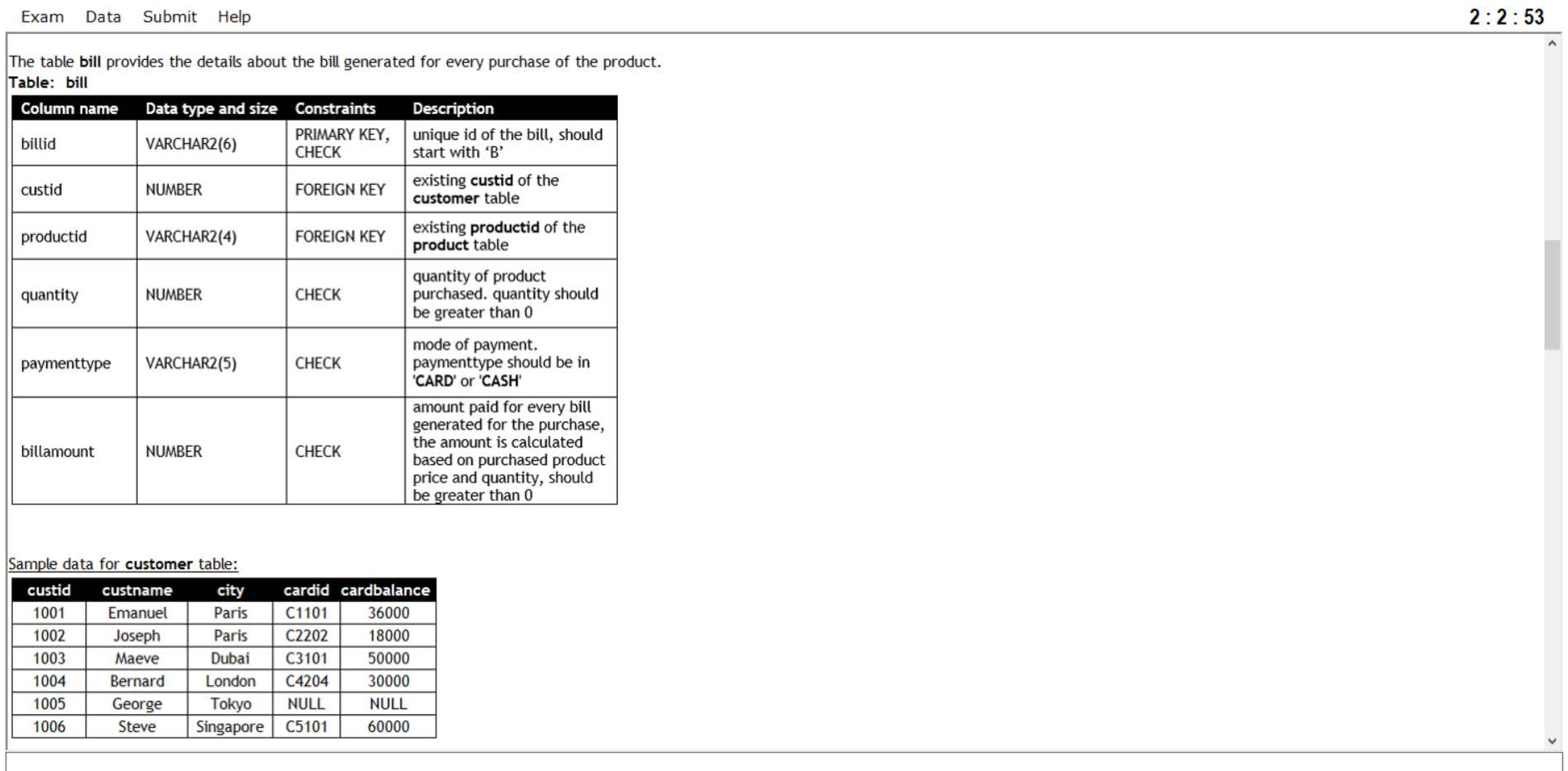
**Table: product**

| Column name | Data type and size | Constraints | Description |
|---|---|---|---|
| productid | VARCHAR2(4) | PRIMARY KEY, CHECK | unique id of the product, should start with 'P' |
| productname | VARCHAR2(20) | NOT NULL | name of the product |
| availableqty | NUMBER | NOT NULL, CHECK | quantity of the available product, available quantity should be greater than or equal to 0 |
| productcost | NUMBER | CHECK | cost of the product, it should be greater than 0 |

The table **bill** provides the details about the bill generated for every purchase of the product.

**Table: bill**

| Column name | Data type and size | Constraints | Description |
|---|---|---|---|
| billid | VARCHAR2(6) | PRIMARY KEY, CHECK | unique id of the bill, should start with 'B' |
| custid | NUMBER | FOREIGN KEY | existing **custid** of the **customer** table |
| productid | VARCHAR2(4) | FOREIGN KEY | existing **productid** of the **product** table |



The table **bill** provides the details about the bill generated for every purchase of the product.

**Table: bill**

| Column name | Data type and size | Constraints | Description |
|---|---|---|---|
| billid | VARCHAR2(6) | PRIMARY KEY, CHECK | unique id of the bill, should start with 'B' |
| custid | NUMBER | FOREIGN KEY | existing **custid** of the **customer** table |
| productid | VARCHAR2(4) | FOREIGN KEY | existing **productid** of the **product** table |
| quantity | NUMBER | CHECK | quantity of product purchased. quantity should be greater than 0 |
| paymenttype | VARCHAR2(5) | CHECK | mode of payment. paymenttype should be in '**CARD**' or '**CASH**' |
| billamount | NUMBER | CHECK | amount paid for every bill generated for the purchase, the amount is calculated based on purchased product price and quantity, should be greater than 0 |

Sample data for **customer** table:

| custid | custname | city | cardid | cardbalance |
|---|---|---|---|---|
| 1001 | Emanuel | Paris | C1101 | 36000 |
| 1002 | Joseph | Paris | C2202 | 18000 |
| 1003 | Maeve | Dubai | C3101 | 50000 |
| 1004 | Bernard | London | C4204 | 30000 |
| 1005 | George | Tokyo | NULL | NULL |
| 1006 | Steve | Singapore | C5101 | 60000 |

Exam Data Submit Help 2 : 2 : 41

Sample data for **product** table:

| productid | productname | availableqty | productcost |
|---|---|---|---|
| P101 | Perfume | 40 | 145 |
| P102 | Shampoo | 35 | 90 |
| P103 | Talc | 28 | 80 |
| P104 | Soap | 25 | 40 |
| P105 | Cold Creams | 15 | 90 |
| P106 | Lip Gloss | 15 | 80 |

Sample data for **bill** table:

| billid | custid | productid | quantity | paymenttype | billamount |
|---|---|---|---|---|---|
| B1001 | 1001 | P102 | 4 | CARD | 360 |
| B1002 | 1001 | P103 | 5 | CASH | 400 |
| B1003 | 1003 | P103 | 6 | CASH | 480 |
| B1004 | 1004 | P103 | 2 | CARD | 160 |
| B1005 | 1002 | P101 | 2 | CASH | 290 |
| B1006 | 1004 | P105 | 4 | CASH | 360 |
| B1007 | 1001 | P102 | 4 | CASH | 360 |
| B1008 | 1004 | P105 | 4 | CASH | 360 |

**Write SQL Queries for the following problems: [20 marks]**

Important Note: For the given requirements, display UNIQUE records wherever applicable.

A. Display **custid**, **custname** and **cardbalance** of the customer(s) having 2 occurrence of letter ‘e’ anywhere in their name and have a card balance of 50000 or less.
Perform the case **insensitive** search.
For the given sample data, the following is the expected output.

| CUSTID | CUSTNAME | CARDBALANCE |
|---|---|---|
| 1001 | Emanuel | 36000 |
| 1003 | Maeve | 50000 |

Exam Data Submit Help 2 : 2 : 31

**Note**: Type the solution in ***dbms_solA.sql*** file. **[2 marks]**

B. Display **custid** and total *count* of product(s) purchased by the customer as "**PRODUCTSCOUNT**" (column alias) with bill amount more than 300 for each generated bill.
For the given sample data, the following records are part of the output along with other record(s).

| CUSTID | PRODUCTSCOUNT |
|---|---|
| 1003 | 1 |
| 1001 | 3 |

**Note**: Type the solution in ***dbms_solB.sql*** file. **[2 marks]**

C. Display **custname** and **city** of the customer(s) who do not have a card and have not purchased any products.
For the given sample data, the following is the expected output.

| CUSTNAME | CITY |
|---|---|
| George | Tokyo |

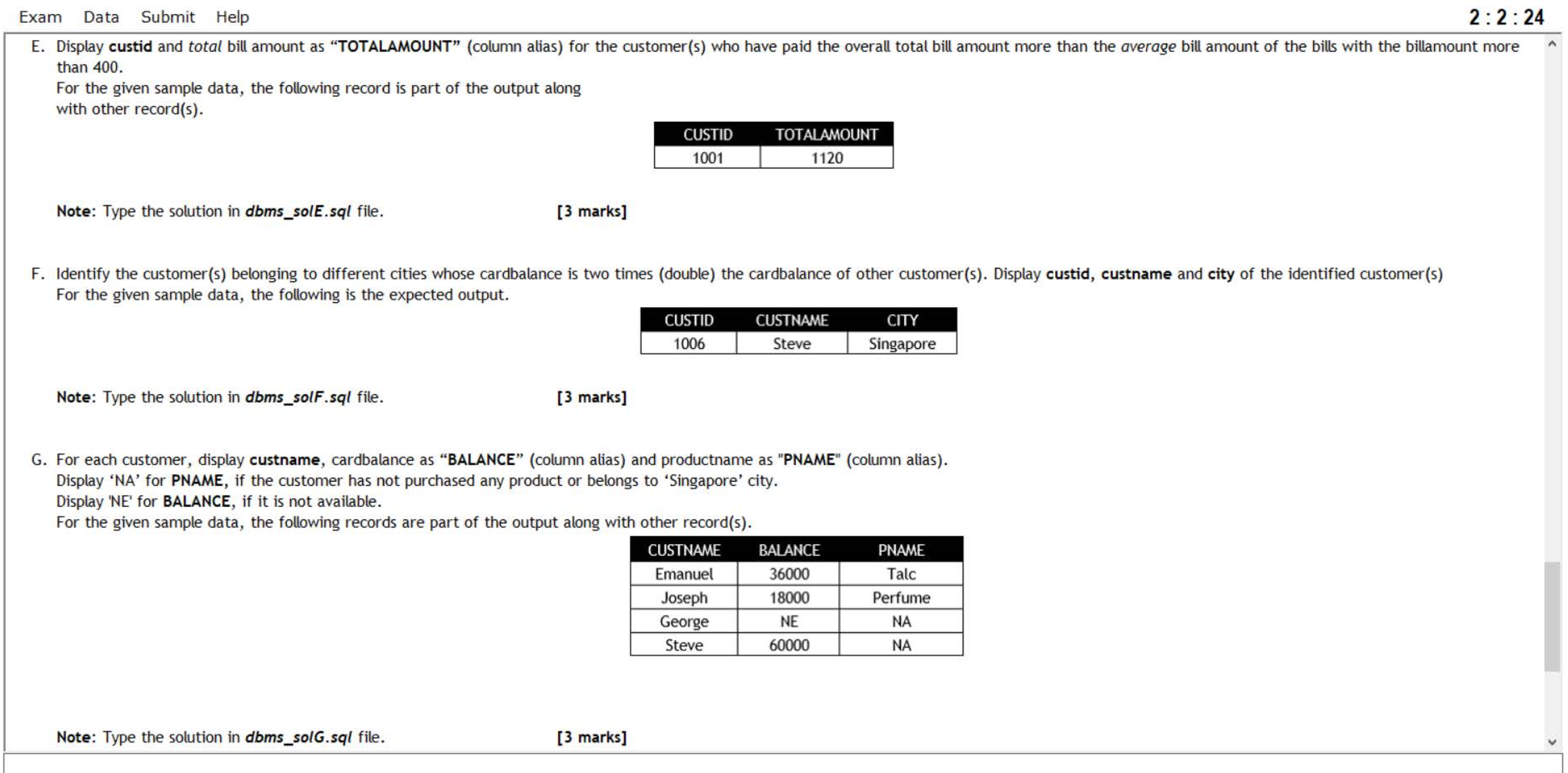
**Note**: Type the solution in ***dbms_solC.sql*** file. **[2 marks]**

D. Display **productid** and **productname** for the product(s) which are purchased by the customer and amount is paid through 'CARD'.
For the given sample data, the following is the expected output.

| PRODUCTID | PRODUCTNAME |
|---|---|
| P102 | Shampoo |
| P103 | Talc |

**Note**: Type the solution in ***dbms_solD.sql*** file. **[2 marks]**

E. Display **custid** and *total* bill amount as "**TOTALAMOUNT**" (column alias) for the customer(s) who have paid the overall total bill amount more than the *average* bill amount of the bills with the billamount more than 400.
For the given sample data, the following record is part of the output along
with other record(s).



E. Display **custid** and *total* bill amount as **"TOTALAMOUNT"** (column alias) for the customer(s) who have paid the overall total bill amount more than the *average* bill amount of the bills with the billamount more than 400.
For the given sample data, the following record is part of the output along with other record(s).

| CUSTID | TOTALAMOUNT |
|---|---|
| 1001 | 1120 |

**Note**: Type the solution in ***dbms_solE.sql*** file. **[3 marks]**

F. Identify the customer(s) belonging to different cities whose cardbalance is two times (double) the cardbalance of other customer(s). Display **custid, custname** and **city** of the identified customer(s)
For the given sample data, the following is the expected output.

| CUSTID | CUSTNAME | CITY |
|---|---|---|
| 1006 | Steve | Singapore |

**Note**: Type the solution in ***dbms_solF.sql*** file. **[3 marks]**

G. For each customer, display **custname**, cardbalance as **"BALANCE"** (column alias) and productname as **"PNAME"** (column alias).
Display 'NA' for **PNAME,** if the customer has not purchased any product or belongs to 'Singapore' city.
Display 'NE' for **BALANCE**, if it is not available.
For the given sample data, the following records are part of the output along with other record(s).

| CUSTNAME | BALANCE | PNAME |
|---|---|---|
| Emanuel | 36000 | Talc |
| Joseph | 18000 | Perfume |
| George | NE | NA |
| Steve | 60000 | NA |

**Note**: Type the solution in ***dbms_solG.sql*** file. **[3 marks]**

G. For each customer, display **custname**, cardbalance as "**BALANCE**" (column alias) and productname as "**PNAME**" (column alias).
Display 'NA' for **PNAME**, if the customer has not purchased any product or belongs to 'Singapore' city.
Display 'NE' for **BALANCE**, if it is not available.
For the given sample data, the following records are part of the output along with other record(s).

| CUSTNAME | BALANCE | PNAME |
|---|---|---|
| Emanuel | 36000 | Talc |
| Joseph | 18000 | Perfume |
| George | NE | NA |
| Steve | 60000 | NA |

**Note**: Type the solution in ***dbms_solG.sql*** file. **[3 marks]**

H. Display **billid**, **custid** and **productid** of the bill(s) which are generated for the purchases done by the customer(s) who are from 'Paris' for the product(s) that are available with quantity more than 30.
For the given sample data, the following records are part of the output along with other record(s).

| BILLID | CUSTID | PRODUCTID |
|---|---|---|
| B1005 | 1002 | P101 |
| B1001 | 1001 | P102 |

Write the query using **Correlated subquery** concept only.

**Note**: Type the solution in ***dbms_solH.sql*** file. **[3 marks]**

```sql
SELECT custid, custname, cardbalance FROM customer WHERE lower(custname) LIKE '%e%e%' AND cardbalance <= 50000;

SELECT CUSTID, COUNT(PRODUCTID) AS "PRODUCTSCOUNT" FROM BILL WHERE BILLAMOUNT >300 GROUP BY CUSTID;

SELECT CUSTNAME, CITY FROM CUSTOMER WHERE CARDID IS NULL AND CUSTID NOT IN (SELECT DISTINCT CUSTID FROM BILL);

SELECT PRODUCTID, PRODUCTNAME FROM PRODUCT WHERE PRODUCTID IN (SELECT DISTINCT PRODUCTID FROM BILL WHERE PAYMENTTYPE = 'CARD');

SELECT B1.CUSTID, SUM(B1.BILLAMOUNT) AS "TOTALAMOUNT" FROM BILL B1 GROUP BY B1.CUSTID HAVING SUM(B1.BILLAMOUNT) > (SELECT AVG(B2.BILLAMOUNT) FROM BILL B2 WHERE B2.BILLAMOUNT

SELECT C1.CUSTNAME, C1.CUSTID, C1.CITY FROM CUSTOMER C1 INNER JOIN CUSTOMER C2 ON C1.CITY <> C2. CITY AND C1.CARDBALANCE = 2*C2.CARDBALANCE;

SELECT DISTINCT CUSTNAME, NVL(TO_CHAR(CARDBALANCE), 'NE') AS "BALANCE", NVL(PRODUCTNAME, 'NA') AS "PNAME" FROM CUSTOMER C
        LEFT OUTER JOIN BILL B ON C.CUSTID = B.CUSTID AND C.CITY <> 'Singapore' LEFT OUTER JOIN
                PRODUCT P ON B.PRODUCTID = P.PRODUCTID;

SELECT BILLID, CUSTID, PRODUCTID FROM BILL B WHERE EXISTS
        (SELECT 1 FROM CUSTOMER C WHERE CITY = 'Paris' AND B.CUSTID = C.CUSTID AND EXISTS
                (SELECT 1 FROM PRODUCT  P WHERE AVAILABLEQTY > 30 AND P.PRODUCTID = B. PRODUCTID));
```