Chapter_2

September 7, 2021

1 Chapter 2 Questions 2.25-2.39 Page 122

[1]:
```
import pymysql.cursors
import pandas as pd

# Connect to the database at 10.1.11.26
connection = pymysql.connect(
    host = "10.1.11.26",
    user = "jtelaak",
    password = "password",
    database = "cape_codd",
    charset = "utf8mb4",
    cursorclass = pymysql.cursors.DictCursor
)
```

QUESTION 2.25: Write an SQL statement to display the SKU, SKU_Description, and WarehouseID for products that have a QuantityOnHand equal to 0. Sort the results in ascending order by WarehouseID.

[2]:
```
sql = "SELECT SKU, SKU_Description, WarehouseID FROM INVENTORY WHERE␣
 ↪QuantityOnHand = 0 ORDER BY WarehouseID ASC ;"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[2]:

	SKU	SKU_Description	WarehouseID
0	101100	Dive Mask, Small Clear	100
1	101100	Dive Mask, Small Clear	200
2	301000	Light Fly Climbing Harness	300
3	201000	Half-dome Tent	400
4	202000	Half-dome Tent Vestibule	400
5	301000	Light Fly Climbing Harness	400
6	302000	Locking Carabiner, Oval	400

QUESTION 2.26: Write an SQL statement to display the SKU, SKU_Description, and WarehouseID for products that have a QuantityOnHand greater than 0. Sort the results in descending order by WarehouseID and in ascending order by SKU.

[3]:
```
sql = "SELECT SKU, SKU_Description, WarehouseID FROM INVENTORY WHERE␣
 ↪QuantityOnHand > 0 ORDER BY SKU ASC, WarehouseID DESC ;"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[3]:

	SKU	SKU_Description	WarehouseID
0	100100	Std. Scuba Tank, Yellow	400
1	100100	Std. Scuba Tank, Yellow	300
2	100100	Std. Scuba Tank, Yellow	200
3	100100	Std. Scuba Tank, Yellow	100
4	100200	Std. Scuba Tank, Magenta	400
5	100200	Std. Scuba Tank, Magenta	300
6	100200	Std. Scuba Tank, Magenta	200
7	100200	Std. Scuba Tank, Magenta	100
8	101100	Dive Mask, Small Clear	400
9	101100	Dive Mask, Small Clear	300
10	101200	Dive Mask, Med Clear	400
11	101200	Dive Mask, Med Clear	300
12	101200	Dive Mask, Med Clear	200
13	101200	Dive Mask, Med Clear	100
14	201000	Half-dome Tent	300
15	201000	Half-dome Tent	200
16	201000	Half-dome Tent	100
17	202000	Half-dome Tent Vestibule	300
18	202000	Half-dome Tent Vestibule	200
19	202000	Half-dome Tent Vestibule	100
20	301000	Light Fly Climbing Harness	200
21	301000	Light Fly Climbing Harness	100
22	302000	Locking Carabiner, Oval	300
23	302000	Locking Carabiner, Oval	200
24	302000	Locking Carabiner, Oval	100

QUESTION 2.27: Write an SQL statement to display SKU, SKU_Description, and WarehouseID for all products that have a QuantityOnHand equal to 0 and a QuantityOnOrder greater than 0. Sort the results in descending order by WarehouseID and in ascending order by SKU.

[4]:
```
sql = ("SELECT SKU, SKU_Description, WarehouseID FROM INVENTORY WHERE␣
 ↪QuantityOnHand = 0 AND QuantityOnOrder > 0 " +
    "ORDER BY WarehouseID DESC, SKU ASC ;")

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[4]:

	SKU	SKU_Description	WarehouseID
0	201000	Half-dome Tent	400
1	202000	Half-dome Tent Vestibule	400

```
2  301000  Light Fly Climbing Harness          400
3  302000     Locking Carabiner, Oval          400
4  301000  Light Fly Climbing Harness          300
5  101100      Dive Mask, Small Clear          200
6  101100      Dive Mask, Small Clear          100
```

QUESTION 2.28: Write an SQL statement to display SKU, SKU_Description, and WarehouseID for all products that have a QuantityOnHand equal to 0 or a QuantityOnOrder equal to 0. Sort the results in descending order by WarehouseID and in ascending order by SKU.

[5]:
```
sql = ("SELECT SKU, SKU_Description, WarehouseID FROM INVENTORY WHERE␣
 ↪QuantityOnHand = 0 AND QuantityOnOrder = 0 " +
"ORDER BY WarehouseID DESC, SKU ASC ; ")

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[5]:
```
Empty DataFrame
Columns: [SKU, SKU_Description, WarehouseID]
Index: []
```

Question 2.29: Write an SQL statement to display the SKU, SKU_Description, WarehouseID, and QuantityOnHand for all products having a QuantityOnHand greater than 1 and less than 10. Do not use the BETWEEN keyword.

[6]:
```
sql = ("SELECT SKU, SKU_Description, WarehouseID, QuantityOnHand FROM INVENTORY␣
 ↪" +
       " WHERE QuantityOnHand > 1 AND QuantityOnHand < 10 ;")

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[6]:
```
      SKU SKU_Description  WarehouseID  QuantityOnHand
0  201000  Half-dome Tent          100               2
```

QUESTION 2.30: Write an SQL statement to display the SKU, SKU_Description, WarehouseID, and QuantityOnHand for all products having a QuantityOnHand greater than 1 and less than 10. Use the BETWEEN keyword.

[7]:
```
sql = "SELECT SKU, SKU_DESCRIPTION, WarehouseID, QuantityOnHand FROM INVENTORY␣
 ↪WHERE QuantityOnHand BETWEEN 1 AND 2"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[7]:
```
      SKU           SKU_DESCRIPTION  WarehouseID  QuantityOnHand
0  201000            Half-dome Tent          100               2
1  202000  Half-dome Tent Vestibule          200               1
```

QUESTION 2.31 Write an SQL statement to show a unique SKU and SKU_Description for all prod- ucts with an SKU description starting with 'Half-Dome'.

[8]:
```
sql = "SELECT DISTINCT SKU, SKU_Description FROM INVENTORY WHERE␣
 ↪SKU_Description LIKE \"Half-Dome%\" ;"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[8]:
```
       SKU           SKU_Description
0   201000            Half-dome Tent
1   202000  Half-dome Tent Vestibule
```

QUESTION 2.32 Write an SQL statement to show a unique SKU and SKU_Description for all prod- ucts with a description that includes the word 'Climb'.

[9]:
```
sql = "SELECT DISTINCT SKU, SKU_Description FROM INVENTORY WHERE␣
 ↪SKU_Description LIKE \"%Climb%\";"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[9]:
```
       SKU             SKU_Description
0   301000  Light Fly Climbing Harness
```

QUESTION 2.33 Write an SQL statement to show a unique SKU and SKU_Description for all prod- ucts with a 'd' in the third position from the left in SKU_Description.

[10]:
```
sql = "SELECT DISTINCT SKU, SKU_Description FROM INVENTORY WHERE␣
 ↪SKU_Description LIKE \"__d%\";"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[10]:
```
       SKU           SKU_Description
0   100100   Std. Scuba Tank, Yellow
1   100200  Std. Scuba Tank, Magenta
```

QUESTION 2.34 Write an SQL statement that uses all of the SQL built-in functions on the Quantity- OnHand column. Include meaningful column names in the result.

[11]:
```
sql = "SELECT SKU, SKU_Description, SUM(QuantityOnHand) AS TotalOnHand,␣
 ↪AVG(QuantityOnHand) AS AvgOnHand, MIN(QuantityOnHand) AS MinOnHand,␣
 ↪MAX(QuantityOnHand)AS MaxOnHand, COUNT(QuantityOnHand) AS TotalWareHouseIDS␣
 ↪FROM INVENTORY GROUP BY SKU_Description, SKU ;"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[11]:

	SKU	SKU_Description	TotalOnHand	AvgOnHand	MinOnHand	MaxOnHand	TotalWareHouseIDS
0	101200	Dive Mask, Med Clear	875.0	218.75	50	475	4
1	101100	Dive Mask, Small Clear	750.0	187.50	0	450	4
2	201000	Half-dome Tent	262.0	65.50	0	250	4
3	202000	Half-dome Tent Vestibule	111.0	27.75	0	100	4
4	301000	Light Fly Climbing Harness	550.0	137.50	0	300	4
5	302000	Locking Carabiner, Oval	2750.0	687.50	0	1250	4
6	100200	Std. Scuba Tank, Magenta	625.0	156.25	75	250	4
7	100100	Std. Scuba Tank, Yellow	650.0	162.50	100	250	4

QUESTION 2.35 Explain the difference between the SQL built-in functions COUNT and SUM.

COUNT counts rows that meet the condition while SUM adds integers in a column.

QUESTION 2.36 Write an SQL statement to display the WarehouseID and the sum of QuantityOn- Hand grouped by WarehouseID. Name the sum TotalItemsOnHand and display the results in descending order of TotalItemsOnHand.

[12]:
```
sql = "SELECT WareHouseID, SUM(QuantityOnHand) AS TotalItemsOnHand FROM␣
 ↪INVENTORY GROUP BY WareHouseID ORDER BY TotalItemsOnHand DESC ;"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[12]:

	WareHouseID	TotalItemsOnHand
0	100	1862.0
1	300	1825.0
2	200	1736.0
3	400	1150.0

QUESTION 2.37 Write an SQL statement to display the WarehouseID and the sum of QuantityOn- Hand grouped by WarehouseID. Omit all SKU items that have three or more itemson hand from the sum, name the sum TotalItemsOnHandLT3, and display the results in descending order of TotalItemsOnHandLT3.

[13]:
```
sql = "SELECT WarehouseID, SUM(QuantityOnHand) as TotalItemsOnHandLT3 FROM␣
 ↪INVENTORY WHERE QuantityOnHand < 3 GROUP BY WarehouseID ORDER BY␣
 ↪TotalItemsOnHandLT3 DESC ;"
```

```
df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[13]:

	WarehouseID	TotalItemsOnHandLT3
0	100	2.0
1	200	1.0
2	300	0.0
3	400	0.0

QUESTION 2.38 Write an SQL statement to display the WarehouseID and the sum of Quantity OnHand grouped by WarehouseID. Omit all SKU items that have three or more items on hand from the sum, and name the sum TotalItemsOnHandLT3. Show the WarehouseID only for warehouses having fewer than two SKUs in their TotalItemsOnHandLT3. Display the results in descending order of TotalItemsOnHandLT3.

[14]:
```
sql = "SELECT WarehouseID, SUM(QuantityOnHand) as TotalItemsOnHandLT3 FROM␣
 ↪INVENTORY WHERE QuantityOnHand < 3 GROUP BY WarehouseID HAVING␣
 ↪TotalItemsOnHandLT3 < 2 ORDER BY TotalItemsOnHandLT3 DESC ;"

df = pd.read_sql_query(sql, connection)
df.tail(1000)
```

[14]:

	WarehouseID	TotalItemsOnHandLT3
0	200	1.0
1	300	0.0
2	400	0.0

QUESTION 2.39 In your answer to Review Question 2.38, was the WHERE clause or the HAVING clause applied first? Why?

The MySQL clause order is FROM, WHERE, SELECT, GROUP BY, HAVING, ORDER BY. WHERE Was applied before HAVING.