| **Assignment Title** | **Warehouse database analysis** |
| --- | --- |
| **Skills take away From This Project** | **python, MySQL** |
| **Domain** | **MySQL** |

**Problem Statement:**

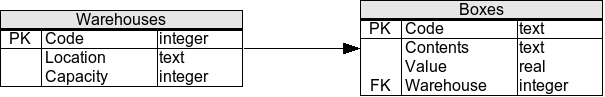
**Warehouse management system is a pivotal part of the supply chain which mainly controls the storage and movement of materials within a warehouse and processes the transactions, including receiving, shipping, picking, and put away. WMS also enables directing and optimizing stock putaway according to the real-time information of bin utilization status. Let’s play with a warehouse dataset provided in the link to answer the following interesting questions.**

**Approach:**

**The SQL file from creating and inserting the data can be found in the following link**

[**https://bit.ly/guvisql1**](https://bit.ly/guvisql1)

**Import or execute the schema in any sql server to build the database and tables it in your local system.The ER diagram for the schema is shown below**

****

**Assignment Questions:**

1. Select all warehouses.
2. Select all boxes with a value larger than $150.
3. Select all distinct contents in all the boxes.
4. Select the average value of all the boxes.
5. Select the warehouse code and the average value of the boxes in each warehouse.
6. Select only those warehouses where the average value of the boxes is greater than 150.
7. Select the code of each box, along with the name of the city the box is located in.
8. Select the warehouse codes, along with the number of boxes in each warehouse.
9. Optionally, take into account that some warehouses are empty (i.e., the box count should show up as zero, instead of omitting the warehouse from the result).
10. Select the codes of all warehouses that are saturated (a warehouse is saturated if the number of boxes in it is larger than the warehouse's capacity).
11. Select the codes of all the boxes located in Chicago.
12. Create a new warehouse in New York with a capacity for 3 boxes.
13. Create a new box, with code "H5RT", containing "Papers" with a value of $200, and located in warehouse 2.
14. Reduce the value of all boxes by 15%.
15. Remove all boxes with a value lower than $100.
16. Remove all boxes from saturated warehouses.
17. Add Index for column "Warehouse" in table "boxes"
    1. -- !!!NOTE!!!: index should NOT be used on small tables in practice
18. Print all the existing indexes
    1. -- !!!NOTE!!!: index should NOT be used on small tables in practice
19. Remove (drop) the index you added just
    1. -- !!!NOTE!!!: index should NOT be used on small tables in practice

**Project Submission:**

**Submit the Assignment notebook**

**(titled with <your name + assignment name>)through github.**

**Project Evaluation metrics:**

* You are supposed to write code in a modular fashion (**in functional blocks**)
* Maintainable: It can be maintained, even as your codebase grows.
* Portable: It works the same in every environment (operating system)
* You have to maintain your code on **GitHub**.(Mandatory)
* You have to keep your **GitHub** repo public so that anyone can check your code.(Mandatory)
* Proper readme file you have to maintain for any project development(Mandatory)
* You should include basic workflow and execution of the entire project in the readme file on **GitHub**
* Follow the coding standards: <https://www.python.org/dev/peps/pep-0008/>
* You need to Create a Demo video of your working model and post in **LinkedIn**(Mandatory)

-- The Warehouse

--3.1

-- Select all warehouses.

select \* from warehouses;

--3.2

-- Select all boxes with a value larger than $150.

select \* from boxes where Value>150;

--3.3

-- Select all distinct contents in all the boxes.

select distinct contents from boxes;

--3.4

-- Select the average value of all the boxes.

select avg(value) from boxes;

--3.5

-- Select the warehouse code and the average value of the boxes in each warehouse.

select warehouse, avg(value) from boxes group by warehouse;

SELECT Warehouse, AVG(Value)

FROM Boxes

GROUP BY Warehouse;

--3.6

-- Same as previous exercise, but select only those warehouses where the average value of the boxes is greater than 150.

select warehouse, avg(value)

from boxes

group by warehouse

having avg(value)> 150;

--3.7

-- Select the code of each box, along with the name of the city the box is located in.

select boxes.code, warehouses.location

from boxes join warehouses

on boxes.Warehouse = Warehouses.Code;

SELECT Boxes.Code, Location

FROM Warehouses

INNER JOIN Boxes ON Warehouses.Code = Boxes.Warehouse;

--3.8

-- Select the warehouse codes, along with the number of boxes in each warehouse.

-- Optionally, take into account that some warehouses are empty (i.e., the box count should show up as zero, instead of omitting the warehouse from the result).

select Warehouse, count(\*)

from boxes

group by warehouse;

--3.9

-- Select the codes of all warehouses that are saturated (a warehouse is saturated if the number of boxes in it is larger than the warehouse's capacity).

select Code

from warehouses join (select warehouse temp\_a, count(\*) temp\_b from boxes group by warehouse) temp

on (warehouses.code = temp.temp\_a)

where warehouses.Capacity<temp.temp\_b;

SELECT Code

FROM Warehouses

WHERE Capacity <

(

SELECT COUNT(\*)

FROM Boxes

WHERE Warehouse = Warehouses.Code

);

--3.10

-- Select the codes of all the boxes located in Chicago.

select Boxes.code

from boxes join Warehouses

on boxes.warehouse = warehouses.code

where warehouses.location = 'Chicago';

/\* Without subqueries \*/

SELECT Boxes.Code

FROM Warehouses LEFT JOIN Boxes

ON Warehouses.Code = Boxes.Warehouse

WHERE Location = 'Chicago';

/\* With a subquery \*/

SELECT Code

FROM Boxes

WHERE Warehouse IN

(

SELECT Code

FROM Warehouses

WHERE Location = 'Chicago'

);

--3.11

-- Create a new warehouse in New York with a capacity for 3 boxes.

INSERT INTO Warehouses VALUES (6, 'New York', 3);

--3.12

-- Create a new box, with code "H5RT", containing "Papers" with a value of $200, and located in warehouse 2.

INSERT INTO Boxes VALUES('H5RT', 'Papers', 200, 2);

--3.13

-- Reduce the value of all boxes by 15%.

update boxes

set value = value \* 0.85;

--3.14

-- Remove all boxes with a value lower than $100.

delete from boxes

where value < 100;

-- 3.15

-- Remove all boxes from saturated warehouses.

delete from boxes

where warehouse in

(

SELECT Code

FROM Warehouses

WHERE Capacity <

(

SELECT COUNT(\*)

FROM Boxes

WHERE Warehouse = Warehouses.Code

)

);

-- 3.16

-- Add Index for column "Warehouse" in table "boxes"

-- !!!NOTE!!!: index should NOT be used on small tables in practice

CREATE INDEX INDEX\_WAREHOUSE ON Boxes (warehouse);

-- 3.17

-- Print all the existing indexes

-- !!!NOTE!!!: index should NOT be used on small tables in practice

-- MySQL

SHOW INDEX FROM Boxes FROM mydb;

SHOW INDEX FROM mydb.Boxes;

-- SQLite

.indexes Boxes

-- OR

SELECT \* FROM SQLITE\_MASTER WHERE type = "index";

-- Oracle

select INDEX\_NAME, TABLE\_NAME, TABLE\_OWNER

from SYS.ALL\_INDEXES

order by TABLE\_OWNER, TABLE\_NAME, INDEX\_NAME

-- 3.18

-- Remove (drop) the index you added just

-- !!!NOTE!!!: index should NOT be used on small tables in practice

DROP INDEX INDEX\_WAREHOUSE;