Overview: We will continue scanning our parts from Demo 1, so we will be focusing on gathering parts from orders 918-932. As shown in **Image 1**, we will be taking parts from their staging shelves into the comp tote. Once all the parts have been gathered inside the comp tote, they will be given to the internet shipper.

Video Link: https://youtu.be/2Gb8JoRZysY

Demo Summary: Once all the parts for an order have been staged, then they will be ready to be picked **(Image 2)**. From here, we are giving a list of orders we need to gather parts from the shelf for. Since we want to continue with the orders we started with in demo 1, we'll go to the end of the page by starting on order 918. From here, the worker will scan the box where these parts will be gathered from. Note, that only comp boxes that are empty or have parts for that same order can be used. In this case, we will use tote 'ORDER_TOTE_00218' to gather our parts. We will then go to our stage locations and scan all of our parts. Similar to the past demos, we can input all the quantities for a part or scan one at a time. We will continue doing this for the rest of the demo.
**Image 3** and **Image 4** shows you the final location for our parts.

End Note: Once the internet shipper gets their order tote, then it's their duty to confirm all the parts. This demo is to assume that it's being run on a Mobile Computer handheld device. This confirmation will have to take place on a desktop.

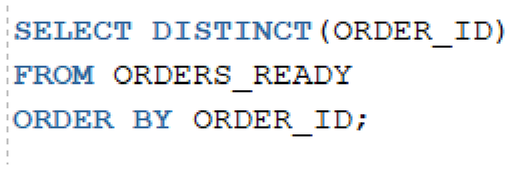
# Code Used

**Python File Used: GUI_Gathering_Parts.py**

**Timestamp: 00:04**
**Python:**

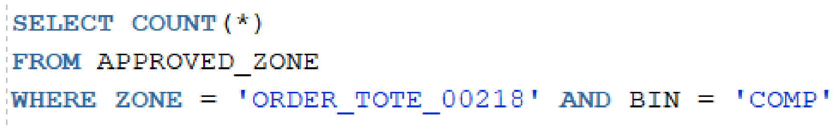
- **load_gather_parts_view(self)**
- **display_order_list(self)**
- **display_extra_order(self)**

**Query 1:** This will display all the orders on the screen. It also allows the worker to go through the list by pressing ‘f5’ or ‘f7’ to move to the next or previous page.



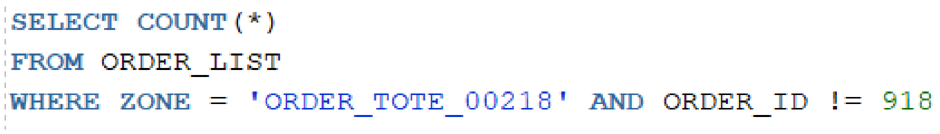
```sql
SELECT DISTINCT(ORDER_ID)
FROM ORDERS_READY
ORDER BY ORDER_ID;
```

**Timestamp: 00:23**
**Python: check_input_box(self, event)**
**Query 2:** This will check if the inputted tote is a ‘COMP’ type tote. We only want to scan our parts to a ‘COMP’ tote before giving them to the shipper. Also, making sure that the inputted tote is valid



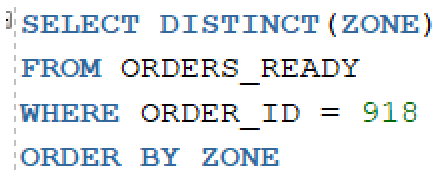
```sql
SELECT COUNT(*)
FROM APPROVED_ZONE
WHERE ZONE = 'ORDER_TOTE_00218' AND BIN = 'COMP'
```

**Python: check_input_box(self, event)**
**Query 3:** This will check if the items, if any, in the tote are part of the same order you are doing. Otherwise, it will reject the ‘COMP’ box



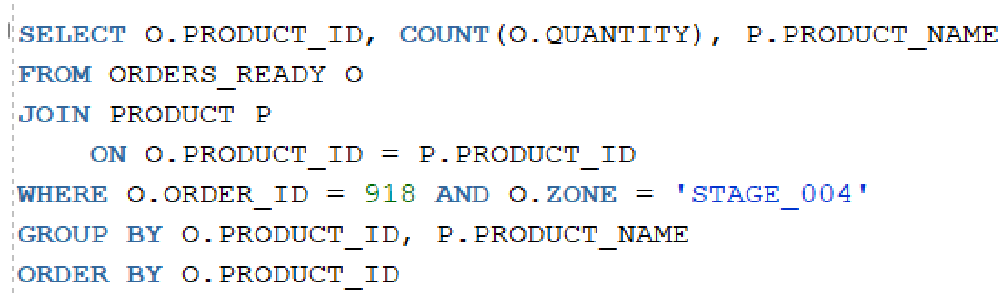
```sql
SELECT COUNT(*)
FROM ORDER_LIST
WHERE ZONE = 'ORDER_TOTE_00218' AND ORDER_ID != 918
```

**Python: load_staging_view(self)**
**Query 4:** This will give us a list of the stages we will be going to. In this case, it will be STAGE_004.



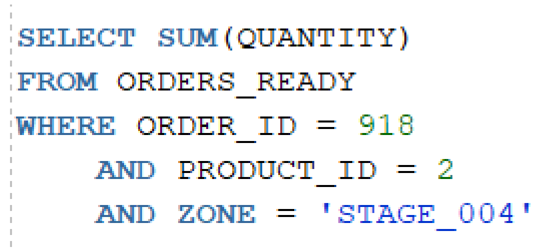
```sql
SELECT DISTINCT(ZONE)
FROM ORDERS_READY
WHERE ORDER_ID = 918
ORDER BY ZONE
```

**Timestamp: 00:27**
**Python: display_product_in_shelf_view(self)**
**Query 5:** This will display all the information about the part you need to scan.



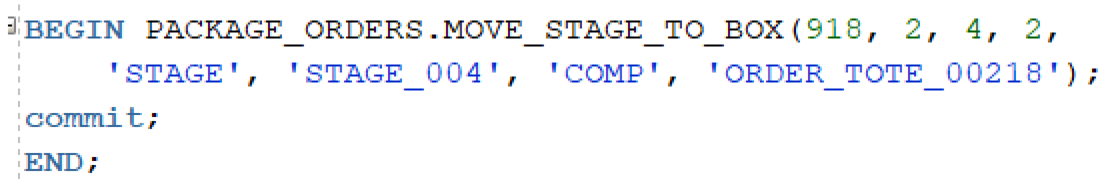
```
SELECT O.PRODUCT_ID, COUNT(O.QUANTITY), P.PRODUCT_NAME
FROM ORDERS_READY O
JOIN PRODUCT P
    ON O.PRODUCT_ID = P.PRODUCT_ID
WHERE O.ORDER_ID = 918 AND O.ZONE = 'STAGE_004'
GROUP BY O.PRODUCT_ID, P.PRODUCT_NAME
ORDER BY O.PRODUCT_ID
```

**Timestamp: 00:29**
**Python: check_product(self, event)**
**Query 6:** This will get the total amount of parts for this shelf. It will then use this number to check if the quantity you inputted is the same or less.



```
SELECT SUM(QUANTITY)
FROM ORDERS_READY
WHERE ORDER_ID = 918
    AND PRODUCT_ID = 2
    AND ZONE = 'STAGE_004'
```

**Python: check_product(self, event)**
**Query 7:** Once all the inputs are approved, then it will move the part or parts to the comp box.



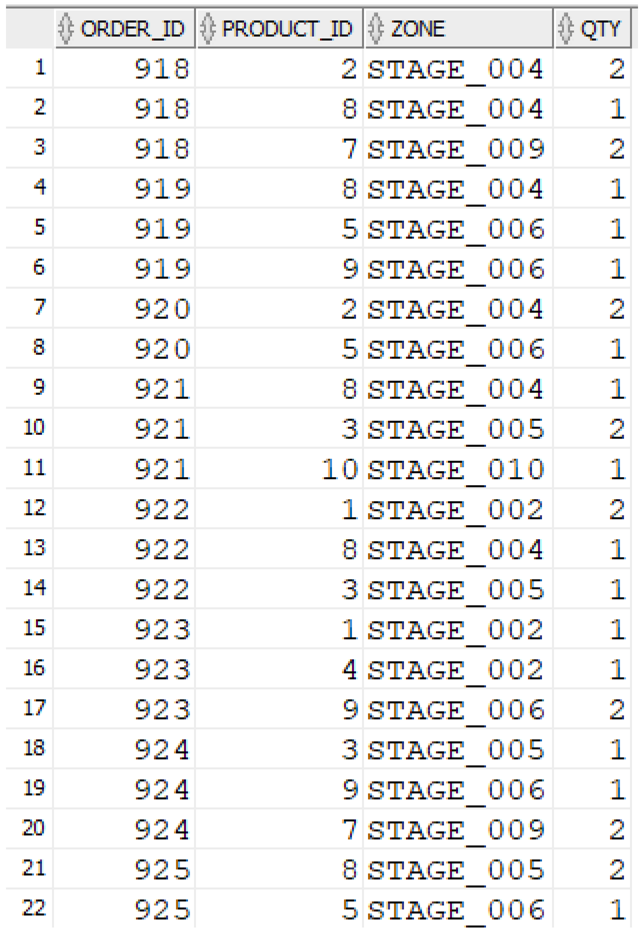
```
BEGIN PACKAGE_ORDERS.MOVE_STAGE_TO_BOX(918, 2, 4, 2,
    'STAGE', 'STAGE_004', 'COMP', 'ORDER_TOTE_00218');
commit;
END;
```

## Extra Notes

**Timestamp 2:48**
In here, I made several mistakes inputting the wrong tote. My query makes sure to tell me that the inputted tote is incorrect and allows me to reenter a new tote.

# Reference Images

**Image 1:** A short view of the location of our parts for order 918-924



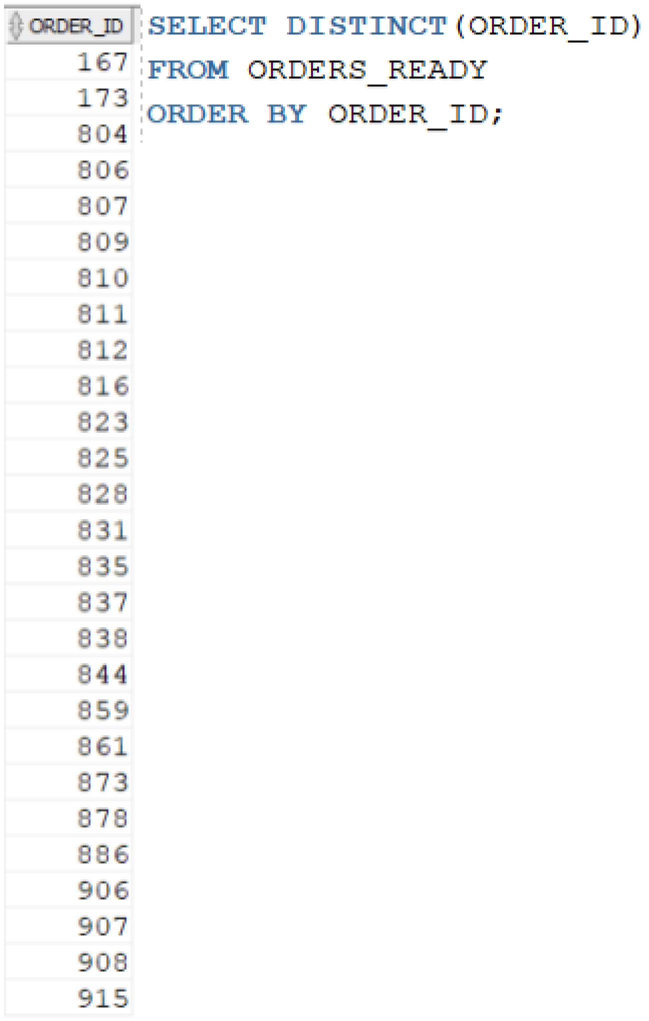
| | ORDER_ID | PRODUCT_ID | ZONE | QTY |
|---|---|---|---|---|
| 1 | 918 | 2 | STAGE_004 | 2 |
| 2 | 918 | 8 | STAGE_004 | 1 |
| 3 | 918 | 7 | STAGE_009 | 2 |
| 4 | 919 | 8 | STAGE_004 | 1 |
| 5 | 919 | 5 | STAGE_006 | 1 |
| 6 | 919 | 9 | STAGE_006 | 1 |
| 7 | 920 | 2 | STAGE_004 | 2 |
| 8 | 920 | 5 | STAGE_006 | 1 |
| 9 | 921 | 8 | STAGE_004 | 1 |
| 10 | 921 | 3 | STAGE_005 | 2 |
| 11 | 921 | 10 | STAGE_010 | 1 |
| 12 | 922 | 1 | STAGE_002 | 2 |
| 13 | 922 | 8 | STAGE_004 | 1 |
| 14 | 922 | 3 | STAGE_005 | 1 |
| 15 | 923 | 1 | STAGE_002 | 1 |
| 16 | 923 | 4 | STAGE_002 | 1 |
| 17 | 923 | 9 | STAGE_006 | 2 |
| 18 | 924 | 3 | STAGE_005 | 1 |
| 19 | 924 | 9 | STAGE_006 | 1 |
| 20 | 924 | 7 | STAGE_009 | 2 |
| 21 | 925 | 8 | STAGE_005 | 2 |
| 22 | 925 | 5 | STAGE_006 | 1 |

**Query Used:**

```sql
SELECT ORDER_ID, PRODUCT_ID, ZONE, SUM(QUANTITY) QTY
FROM ORDERS_READY
WHERE ORDER_ID BETWEEN 918 AND 932
GROUP BY ORDER_ID, PRODUCT_ID, ZONE
ORDER BY ORDER_ID, ZONE, PRODUCT_ID;
```

**Image 2**: This will display all the orders that are ready to be given to the shipper. Note that this is after our demo demonstration. Otherwise, order 918-932 would have appeared here as well. Also, you can also see these orders in the demo as I'm going through the list.

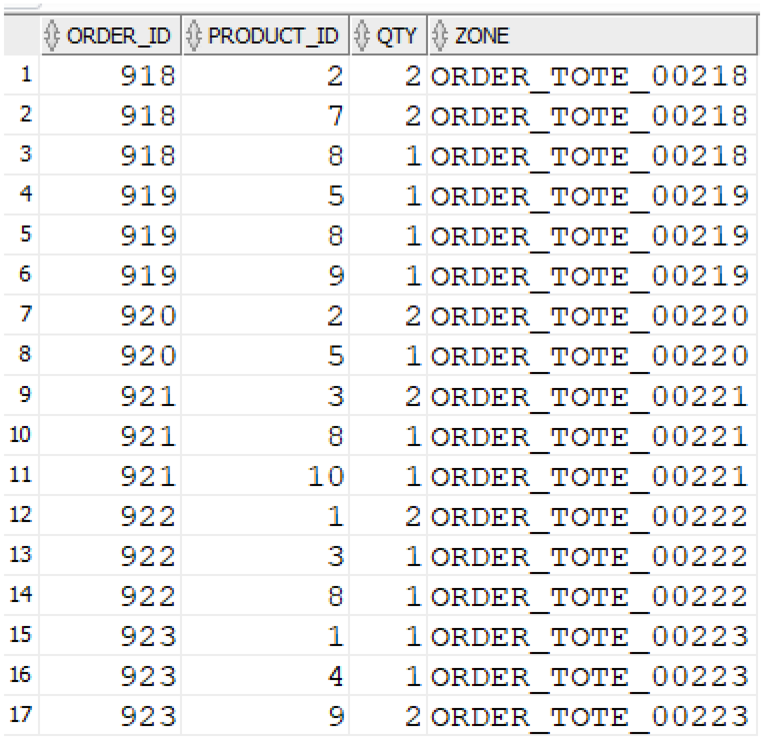
**ORDERS_READY VIEW**
Link:
https://github.com/Cephuez/PastProjects/blob/main/Auto_Parts_Warehouse_Project/1_Auto_Part_Database/Views/ORDERS_READY



```
SELECT DISTINCT(ORDER_ID)
FROM ORDERS_READY
ORDER BY ORDER_ID;
```

```
ORDER_ID
     167
     173
     804
     806
     807
     809
     810
     811
     812
     816
     823
     825
     828
     831
     835
     837
     838
     844
     859
     861
     873
     878
     886
     906
     907
     908
     915
```

**Image 3:** Final update on our parts for this demo. This shows you the location of all the parts for an order.

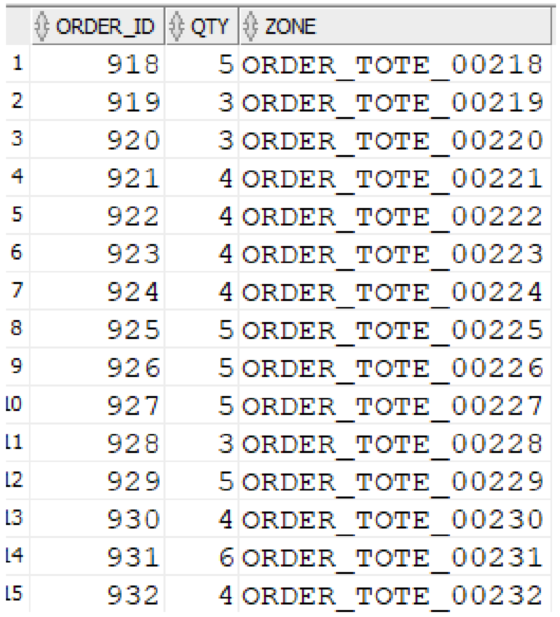


|  | ORDER_ID | PRODUCT_ID | QTY | ZONE |
|---|---|---|---|---|
| 1 | 918 | 2 | 2 | ORDER_TOTE_00218 |
| 2 | 918 | 7 | 2 | ORDER_TOTE_00218 |
| 3 | 918 | 8 | 1 | ORDER_TOTE_00218 |
| 4 | 919 | 5 | 1 | ORDER_TOTE_00219 |
| 5 | 919 | 8 | 1 | ORDER_TOTE_00219 |
| 6 | 919 | 9 | 1 | ORDER_TOTE_00219 |
| 7 | 920 | 2 | 2 | ORDER_TOTE_00220 |
| 8 | 920 | 5 | 1 | ORDER_TOTE_00220 |
| 9 | 921 | 3 | 2 | ORDER_TOTE_00221 |
| 10 | 921 | 8 | 1 | ORDER_TOTE_00221 |
| 11 | 921 | 10 | 1 | ORDER_TOTE_00221 |
| 12 | 922 | 1 | 2 | ORDER_TOTE_00222 |
| 13 | 922 | 3 | 1 | ORDER_TOTE_00222 |
| 14 | 922 | 8 | 1 | ORDER_TOTE_00222 |
| 15 | 923 | 1 | 1 | ORDER_TOTE_00223 |
| 16 | 923 | 4 | 1 | ORDER_TOTE_00223 |
| 17 | 923 | 9 | 2 | ORDER_TOTE_00223 |

**Query Used:**

```
SELECT ORDER_ID, PRODUCT_ID, SUM(UNITS) QTY, ZONE
FROM ORDER_LIST
WHERE ORDER_ID BETWEEN 918 AND 932
GROUP BY ORDER_ID, PRODUCT_ID, ZONE
ORDER BY ORDER_ID;
```

**Image 4:** This demonstrates how many parts for an order are inside each tote.



| | ORDER_ID | QTY | ZONE |
|---|---|---|---|
| 1 | 918 | 5 | ORDER_TOTE_00218 |
| 2 | 919 | 3 | ORDER_TOTE_00219 |
| 3 | 920 | 3 | ORDER_TOTE_00220 |
| 4 | 921 | 4 | ORDER_TOTE_00221 |
| 5 | 922 | 4 | ORDER_TOTE_00222 |
| 6 | 923 | 4 | ORDER_TOTE_00223 |
| 7 | 924 | 4 | ORDER_TOTE_00224 |
| 8 | 925 | 5 | ORDER_TOTE_00225 |
| 9 | 926 | 5 | ORDER_TOTE_00226 |
| 10 | 927 | 5 | ORDER_TOTE_00227 |
| 11 | 928 | 3 | ORDER_TOTE_00228 |
| 12 | 929 | 5 | ORDER_TOTE_00229 |
| 13 | 930 | 4 | ORDER_TOTE_00230 |
| 14 | 931 | 6 | ORDER_TOTE_00231 |
| 15 | 932 | 4 | ORDER_TOTE_00232 |

**Query Used:**

```
SELECT ORDER_ID, SUM(UNITS) QTY, ZONE
FROM ORDER_LIST
WHERE ORDER_ID BETWEEN 918 AND 932
GROUP BY ORDER_ID, ZONE
ORDER BY ORDER_ID;
```