Overview: We will continue from our Demo 1 example. The worker will now start staging their parts picked from bin ‘A’. This will include the 4 totes we used on the last demo(**Image 1**). It’s the worker’s responsibility to find a location to place these items in the staging area.

Note: This will be done virtually, so assume we have all of our parts inside the totes.

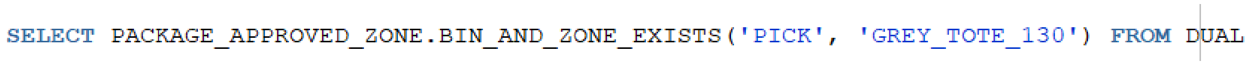
Video Link: https://youtu.be/-sRb5uhVIMs

Demo Summary: The worker will start picking the parts from the tote. We will be dealing with ‘GREY_TOTE_130’ tote first. The worker will then decide which parts are going where. In the first scan, I decided to move 4 parts of brake_pad_1 to STAGE_002. For more flexibility, we have the option of moving parts to whatever staging shelf we want. We will repeat this process until we are done scanning all of our parts from demo 1. **Image 2** displays a small preview of the location of our parts.

Code Used

Python File Used: GUI_Staging_Parts.py

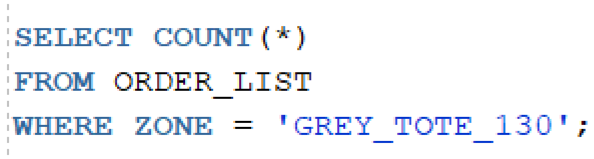
Timestamp: 00:13
Python: check_tote_item_list(self, event)
Query 1: This package checks if the inputted tote is real.



```
SELECT PACKAGE_APPROVED_ZONE.BIN_AND_ZONE_EXISTS('PICK', 'GREY_TOTE_130') FROM DUAL
```

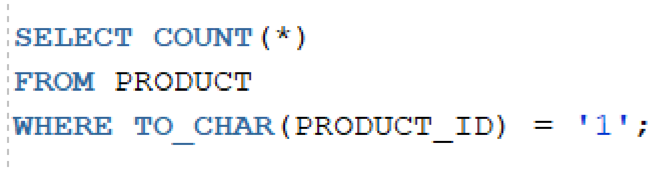
Link:
https://github.com/Cephuez/PastProjects/blob/main/Auto_Parts_Warehouse_Project/1_Auto_Part_Database/Packages/Package%20APPROVED_ZONE

Python: check_tote_item_list(self, event)
Query 2: Once the tote's name has been validated, then it will check if the tote has at least 1 item in it.



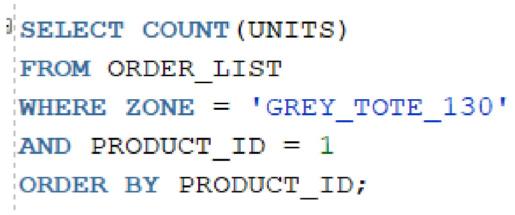
```
SELECT COUNT(*)
FROM ORDER_LIST
WHERE ZONE = 'GREY_TOTE_130';
```

Timestamp: 00:15
Python: check_product_scanned(self, event)
Query 3: We now need to check if the product id inputted is valid.



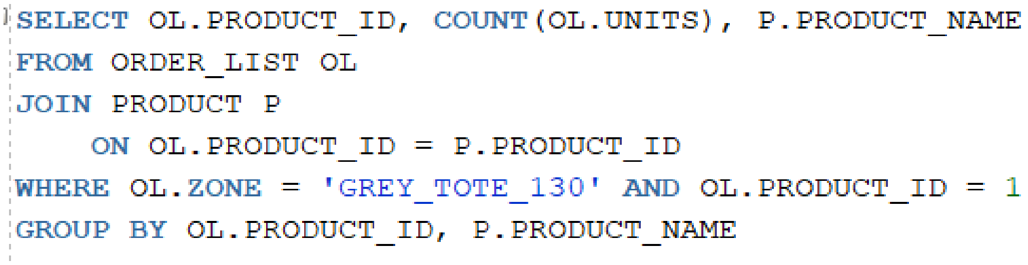
```
SELECT COUNT(*)
FROM PRODUCT
WHERE TO_CHAR(PRODUCT_ID) = '1';
```

Python: check_product_scanned(self, event)
Query 4: Once the inputted product has been validated, then we will check if there is at least 1 of the item in the tote. To be organized, we need to make sure that we are scanning the parts from their appropriate totes.



```
SELECT COUNT(UNITS)
FROM ORDER_LIST
WHERE ZONE = 'GREY_TOTE_130'
AND PRODUCT_ID = 1
ORDER BY PRODUCT_ID;
```

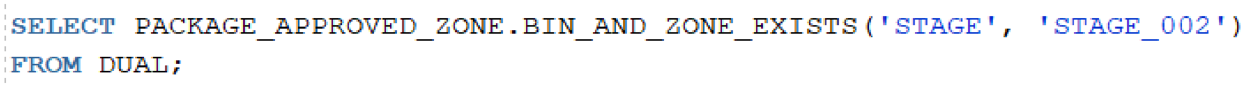
Python: load_product_info(self)
Query 5: The worker will be shown the number of parts that are inside the tote before scanning them to their shelves.



```sql
SELECT OL.PRODUCT_ID, COUNT(OL.UNITS), P.PRODUCT_NAME
FROM ORDER_LIST OL
JOIN PRODUCT P
    ON OL.PRODUCT_ID = P.PRODUCT_ID
WHERE OL.ZONE = 'GREY_TOTE_130' AND OL.PRODUCT_ID = 1
GROUP BY OL.PRODUCT_ID, P.PRODUCT_NAME
```

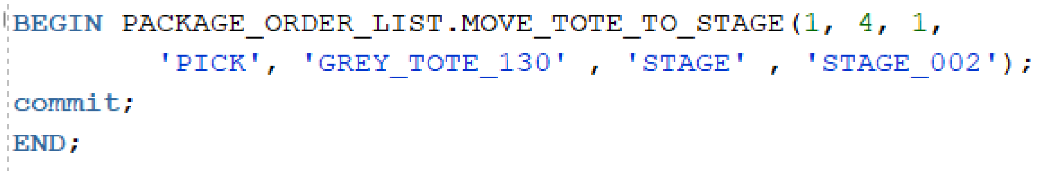
Timestamp: 00:25

Python: stage_part(self, event)
Query 6: Similar to the other query, it will check if the inputted stage name is valid.



```sql
SELECT PACKAGE_APPROVED_ZONE.BIN_AND_ZONE_EXISTS('STAGE', 'STAGE_002')
FROM DUAL;
```

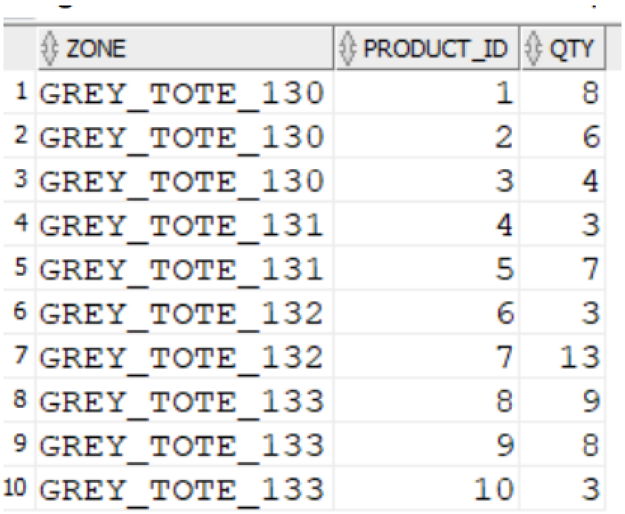
Python: stage_part(self, event)
Query 7: Once all inputs have been approved, it will move the part to its location



```sql
BEGIN PACKAGE_ORDER_LIST.MOVE_TOTE_TO_STAGE(1, 4, 1,
        'PICK', 'GREY_TOTE_130' , 'STAGE' , 'STAGE_002');
commit;
END;
```

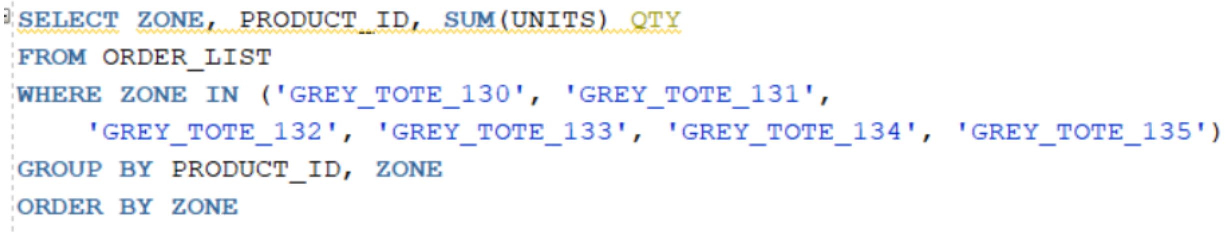
Reference Images

Image 1. We will get our parts from Grey Tote 130 - 133



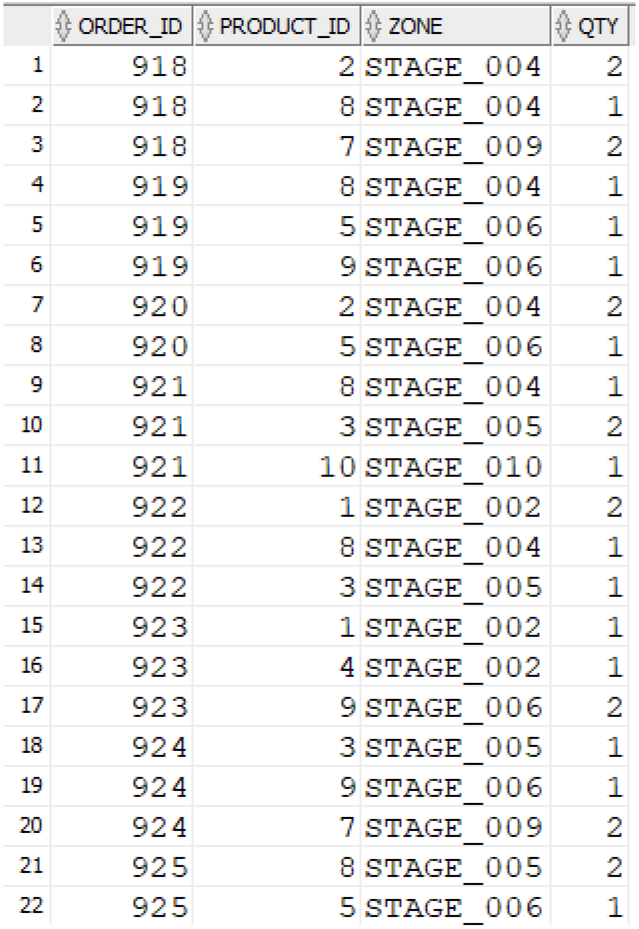
	ZONE	PRODUCT_ID	QTY
1	GREY_TOTE_130	1	8
2	GREY_TOTE_130	2	6
3	GREY_TOTE_130	3	4
4	GREY_TOTE_131	4	3
5	GREY_TOTE_131	5	7
6	GREY_TOTE_132	6	3
7	GREY_TOTE_132	7	13
8	GREY_TOTE_133	8	9
9	GREY_TOTE_133	9	8
10	GREY_TOTE_133	10	3

Query Used



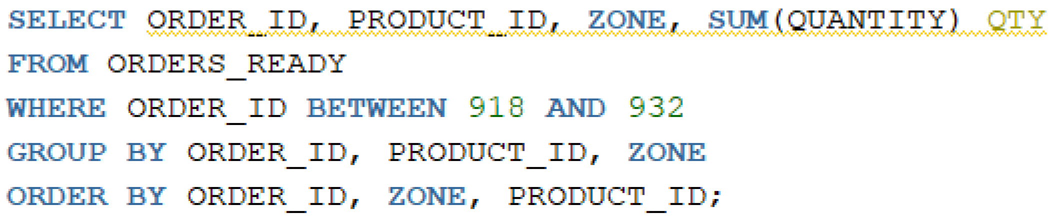
```
SELECT ZONE, PRODUCT_ID, SUM(UNITS) QTY
FROM ORDER_LIST
WHERE ZONE IN ('GREY_TOTE_130', 'GREY_TOTE_131',
    'GREY_TOTE_132', 'GREY_TOTE_133', 'GREY_TOTE_134', 'GREY_TOTE_135')
GROUP BY PRODUCT_ID, ZONE
ORDER BY ZONE
```

Image 2. Displaying the location of some of the parts we scanned.



	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:



```
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;
```