Overview: We will be picking parts that a customer has already confirmed. For this demo, I've generated around 18 orders(918-932) **Image 1**. All of these parts are located in bin 'A' **Image 2**. It's the worker's duty to scan these parts from their shelves and move them to a tote. This tote will contain all the items before being moved to the staging zone.

Note: We will be doing this virtually, so we are assuming that all the parts are in their shelves. Plus, we will be dealing only with parts in Bin 'A' aka Brake Pads only. I've organized my parts by moving them to assigned locations in my warehouse **Image 3**.

Video Link: https://youtu.be/TuX8qcJ62LY

Demo Summary: For the first example, the worker will scan 8 parts of Brake_Pad_1 (Product ID: 1) into a tote. First, I need to make sure that I scan the tote where I will place these items. I press 'F7' to direct us to the scan tote window to scan 'GREY_TOTE_130'. Once we are directed back, we have the option to scan 1 part at a time or more by pressing 'F5'. As the demo continues, I can switch to different totes just in case my current tote has been filled. I will repeat this until we have collected all parts for this zone 'A101'. We then get prompted to move to zone 'A105' and continue our job. Once we are done, we are sent back to the 'Start Picking' window to continue picking other sections. **Image 4** shows you the end result and the current location of the scanned parts.

Code Used

A short list of sql code used throughout the demo. I used python mostly for the GUI construction of this demo and to run functions that read my query to my Oracle Database.

Ex: pd.read_sql(query, self.engine)

- read_sql() function executes my sql query to my database.

Timestamp: 00:34

Query 1: This will display all the bins along with the quantities. The where clause make sure to display parts that have yet to be picked from the shelve, in other words TIME_PICKED is NULL

```
SELECT BIN_LOCATION, SUM(QUANTITY) QUANTITY
FROM PICKS
WHERE TIME_PICKED IS NULL
GROUP BY BIN_LOCATION
ORDER BY BIN_LOCATION
```

Timestamp: 00:36

Query 2: This will give me a list of Zones that have picks in them. Since we are starting by alphabetical order, we will pick the first row of the list returned. In other words, Zone ‘A101’

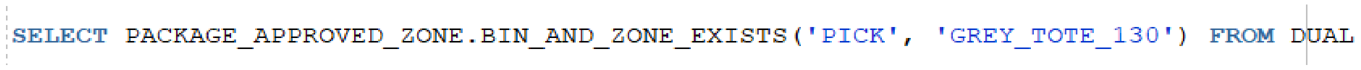
```
SELECT DISTINCT ZONE_lOCATION
FROM PICKS
WHERE BIN_LOCATION = 'A' AND PICK_STATUS = 'N'
ORDER BY ZONE_LOCATION;
```

Timestamp: 00:41

Query 3: This will return me the product id, the quantities needed from this shelf, zone location, and the name of the part.

```
SELECT PK.PRODUCT_ID, SUM(PK.QUANTITY), PK.ZONE_LOCATION, P.PRODUCT_NAME
FROM PICKS PK
JOIN PRODUCT P
    ON PK.PRODUCT_ID = P.PRODUCT_ID
WHERE ZONE_LOCATION = 'A101' AND PICK_STATUS = 'N'
GROUP BY PK.PRODUCT_ID, PK.ZONE_LOCATION, P.PRODUCT_NAME
ORDER BY PK.ZONE LOCATION
```

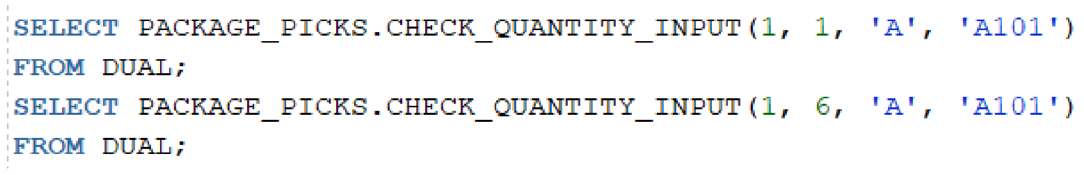
Timestamp: 00:46
Query 4: This procedure returns a number that determines if the tote is valid, otherwise it will prompt the user again.



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

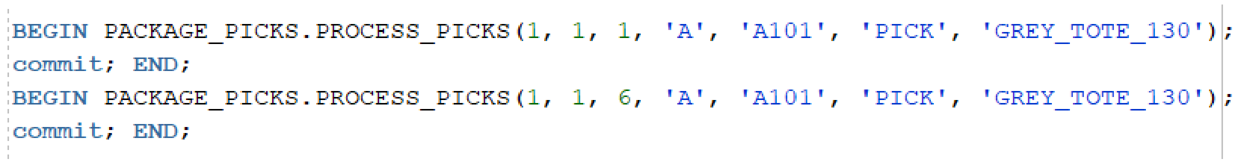
Link to code:
https://github.com/Cephuez/PastProjects/blob/main/Auto_Parts_Warehouse_Project/1_Auto_Part_Database/Packages/Package%20APPROVED_ZONE

Timestamp: 00:55 - 1:06
Query 5: This procedure checks if the quantity inputted isn't more than the amount that needs to be picked.



```
SELECT PACKAGE_PICKS.CHECK_QUANTITY_INPUT(1, 1, 'A', 'A101')
FROM DUAL;
SELECT PACKAGE_PICKS.CHECK_QUANTITY_INPUT(1, 6, 'A', 'A101')
FROM DUAL;
```

Query 6: If all the inputted values are correct, then it will process that pick and move it to its tote location.

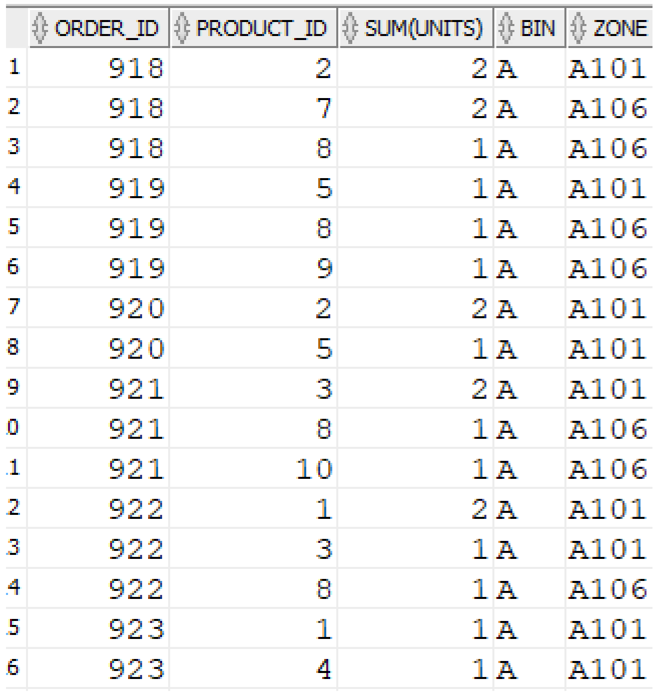


```
BEGIN PACKAGE_PICKS.PROCESS_PICKS(1, 1, 1, 'A', 'A101', 'PICK', 'GREY_TOTE_130');
commit; END;
BEGIN PACKAGE_PICKS.PROCESS_PICKS(1, 1, 6, 'A', 'A101', 'PICK', 'GREY_TOTE_130');
commit; END;
```

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

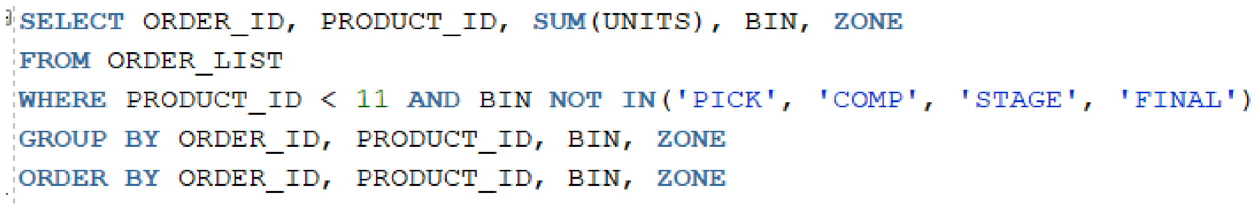
Reference Images

Image 1. A short demonstrations of the parts I need to pick for a few orders



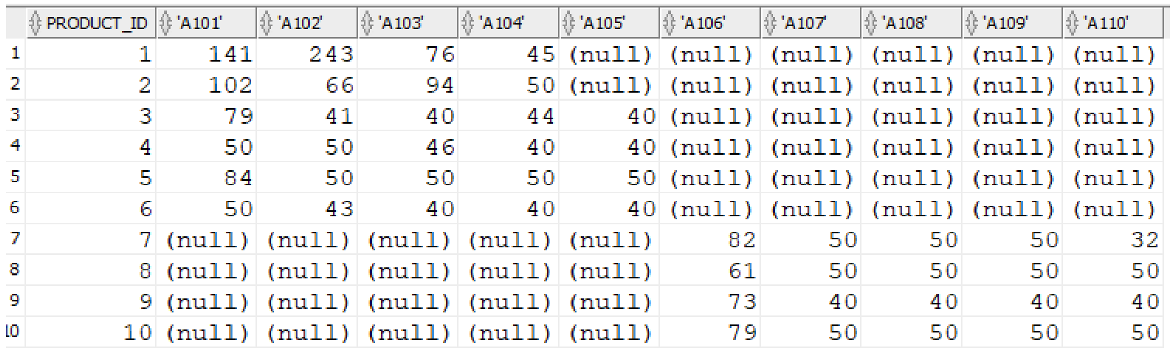
	ORDER_ID	PRODUCT_ID	SUM(UNITS)	BIN	ZONE
1	918	2	2	A	A101
2	918	7	2	A	A106
3	918	8	1	A	A106
4	919	5	1	A	A101
5	919	8	1	A	A106
6	919	9	1	A	A106
7	920	2	2	A	A101
8	920	5	1	A	A101
9	921	3	2	A	A101
0	921	8	1	A	A106
1	921	10	1	A	A106
2	922	1	2	A	A101
3	922	3	1	A	A101
4	922	8	1	A	A106
5	923	1	1	A	A101
6	923	4	1	A	A101

Query Used



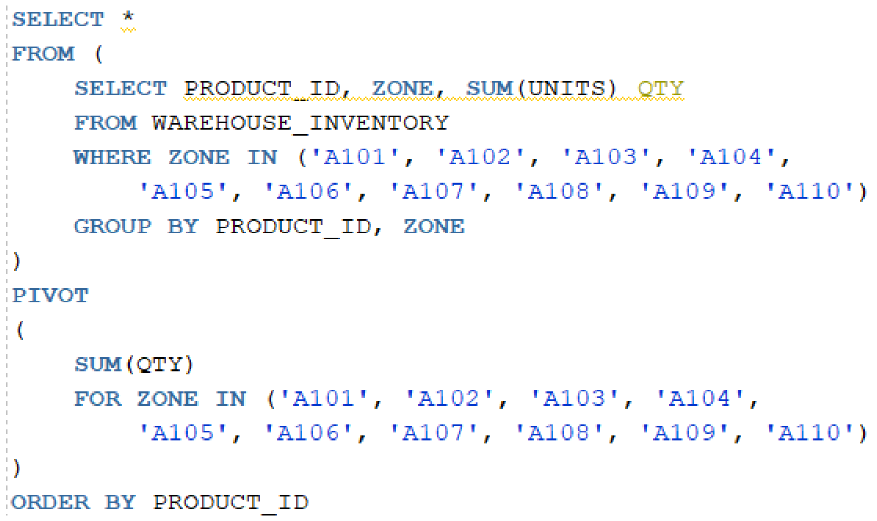
```
SELECT ORDER_ID, PRODUCT_ID, SUM(UNITS), BIN, ZONE
FROM ORDER_LIST
WHERE PRODUCT_ID < 11 AND BIN NOT IN('PICK', 'COMP', 'STAGE', 'FINAL')
GROUP BY ORDER_ID, PRODUCT_ID, BIN, ZONE
ORDER BY ORDER_ID, PRODUCT_ID, BIN, ZONE
```

Image 2. Locations of all brake pads in bin ‘A’. The only products that are brake pads are Product ID 1 - 10



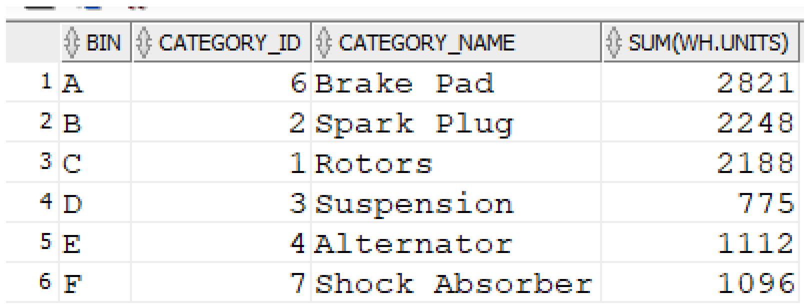
	PRODUCT_ID	'A101'	'A102'	'A103'	'A104'	'A105'	'A106'	'A107'	'A108'	'A109'	'A110'
1	1	141	243	76	45	(null)	(null)	(null)	(null)	(null)	(null)
2	2	102	66	94	50	(null)	(null)	(null)	(null)	(null)	(null)
3	3	79	41	40	44	40	(null)	(null)	(null)	(null)	(null)
4	4	50	50	46	40	40	(null)	(null)	(null)	(null)	(null)
5	5	84	50	50	50	50	(null)	(null)	(null)	(null)	(null)
6	6	50	43	40	40	40	(null)	(null)	(null)	(null)	(null)
7	7	(null)	(null)	(null)	(null)	(null)	82	50	50	50	32
8	8	(null)	(null)	(null)	(null)	(null)	61	50	50	50	50
9	9	(null)	(null)	(null)	(null)	(null)	73	40	40	40	40
10	10	(null)	(null)	(null)	(null)	(null)	79	50	50	50	50

Query Used



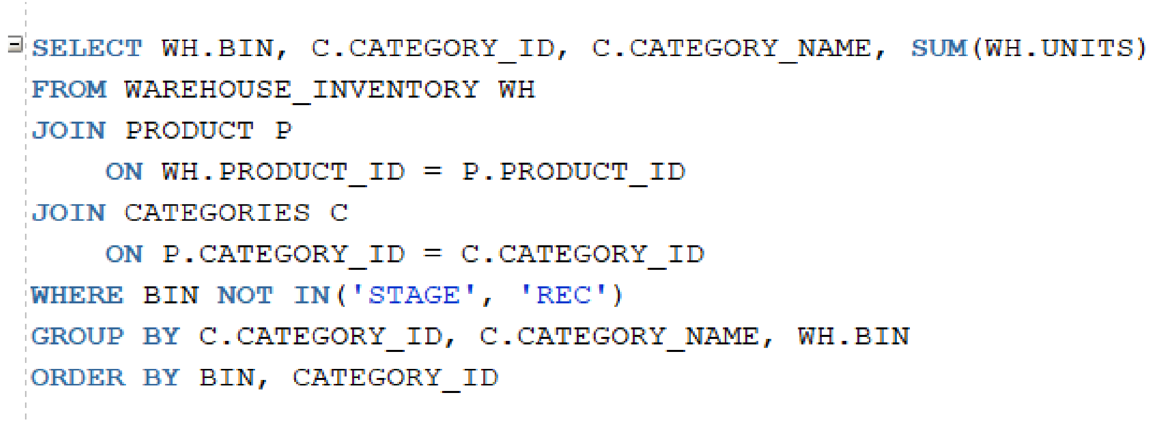
```
SELECT *
FROM (
    SELECT PRODUCT_ID, ZONE, SUM(UNITS) QTY
    FROM WAREHOUSE_INVENTORY
    WHERE ZONE IN ('A101', 'A102', 'A103', 'A104',
        'A105', 'A106', 'A107', 'A108', 'A109', 'A110')
    GROUP BY PRODUCT_ID, ZONE
)
PIVOT
(
    SUM(QTY)
    FOR ZONE IN ('A101', 'A102', 'A103', 'A104',
        'A105', 'A106', 'A107', 'A108', 'A109', 'A110')
)
ORDER BY PRODUCT_ID
```

Image 3. This shows you how my inventory is organized.



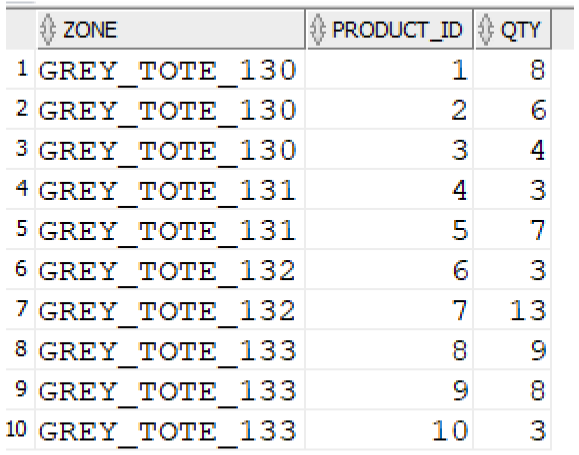
	BIN	CATEGORY_ID	CATEGORY_NAME	SUM(WH.UNITS)
1	A	6	Brake Pad	2821
2	B	2	Spark Plug	2248
3	C	1	Rotors	2188
4	D	3	Suspension	775
5	E	4	Alternator	1112
6	F	7	Shock Absorber	1096

Query Used



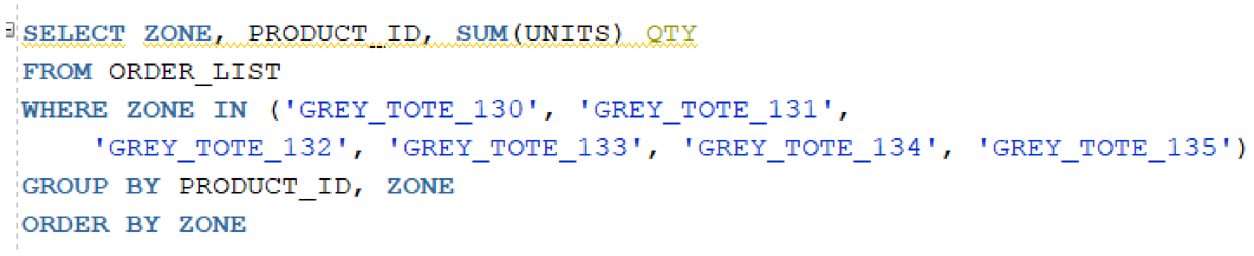
```
SELECT WH.BIN, C.CATEGORY_ID, C.CATEGORY_NAME, SUM(WH.UNITS)
FROM WAREHOUSE_INVENTORY WH
JOIN PRODUCT P
    ON WH.PRODUCT_ID = P.PRODUCT_ID
JOIN CATEGORIES C
    ON P.CATEGORY_ID = C.CATEGORY_ID
WHERE BIN NOT IN('STAGE', 'REC')
GROUP BY C.CATEGORY_ID, C.CATEGORY_NAME, WH.BIN
ORDER BY BIN, CATEGORY_ID
```

Image 4. End Result. The location of our parts after we are done picking them



	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
```