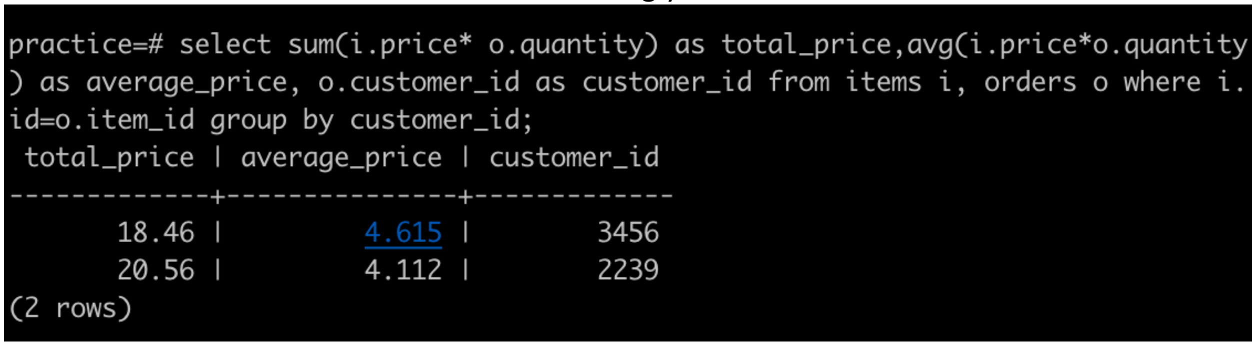
Part 1

1)

select sum(i.price* o.quantity) as total_price,avg(i.price*o.quantity) as average_price, o.customer_id as customer_id from items i, orders o where i.id=o.item_id group by customer_id;

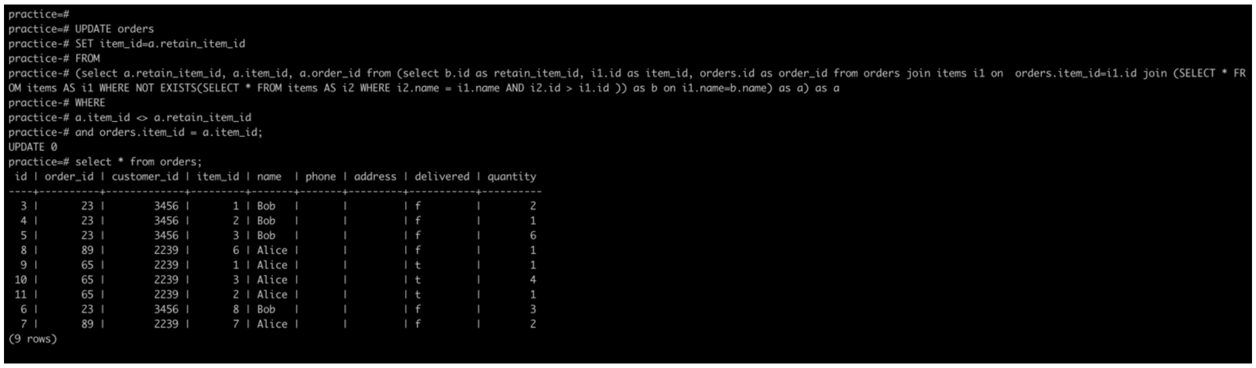
by weight or quantity would have made it more tricky, but this consideration does not matter for our case since cost is also associated accordingly.



```
practice=# select sum(i.price* o.quantity) as total_price,avg(i.price*o.quantity
) as average_price, o.customer_id as customer_id from items i, orders o where i.
id=o.item_id group by customer_id;
 total_price | average_price | customer_id
-------------+---------------+-------------
       18.46 |         4.615 |        3456
       20.56 |         4.112 |        2239
(2 rows)
```

2)

UPDATE orders
SET item_id=a.retain_item_id
FROM
(select a.retain_item_id, a.item_id, a.order_id from (select b.id as retain_item_id, i1.id as item_id, orders.id as order_id from orders join items i1 on orders.item_id=i1.id join (SELECT * FROM items AS i1 WHERE NOT EXISTS(SELECT * FROM items AS i2 WHERE i2.name = i1.name AND i2.id > i1.id)) as b on i1.name=b.name) as a) as a
WHERE
a.item_id <> a.retain_item_id
and orders.item_id = a.item_id;



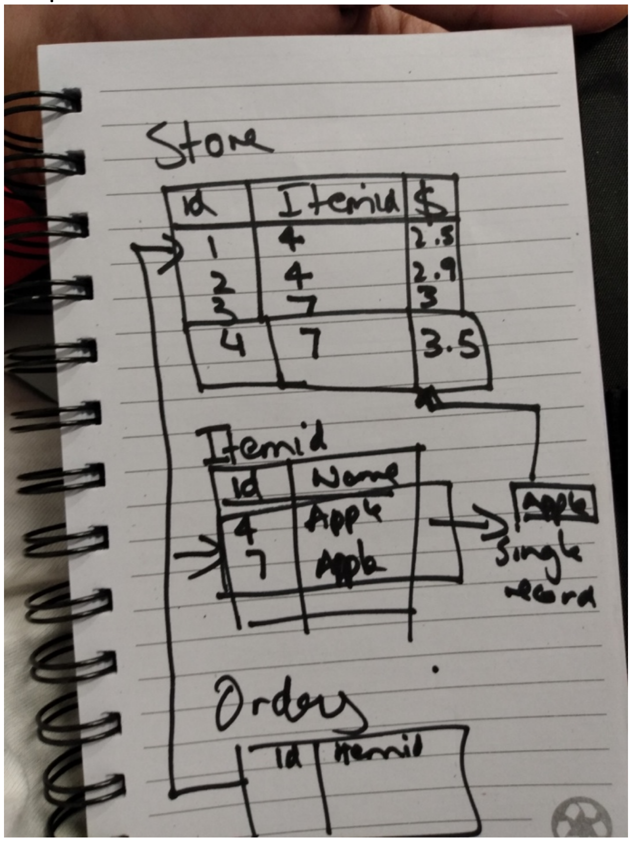
```
practice=#
practice=# UPDATE orders
practice-# SET item_id=a.retain_item_id
practice-# FROM
practice-# (select a.retain_item_id, a.item_id, a.order_id from (select b.id as retain_item_id, i1.id as item_id, orders.id as order_id from orders join items i1 on  orders.item_id=i1.id join (SELECT * FR
OM items AS i1 WHERE NOT EXISTS(SELECT * FROM items AS i2 WHERE i2.name = i1.name AND i2.id > i1.id )) as b on i1.name=b.name) as a) as a
practice-# WHERE
practice-# a.item_id <> a.retain_item_id
practice-# and orders.item_id = a.item_id;
UPDATE 0
practice=# select * from orders;
 id | order_id | customer_id | item_id | name  | phone | address | delivered | quantity
----+----------+-------------+---------+-------+-------+---------+-----------+----------
  3 |       23 |        3456 |       1 | Bob   |       |         | f         |        2
  4 |       23 |        3456 |       2 | Bob   |       |         | f         |        1
  5 |       23 |        3456 |       3 | Bob   |       |         | f         |        6
  8 |       89 |        2239 |       6 | Alice |       |         | f         |        1
  9 |       65 |        2239 |       1 | Alice |       |         | t         |        1
 10 |       65 |        2239 |       3 | Alice |       |         | t         |        4
 11 |       65 |        2239 |       2 | Alice |       |         | t         |        1
  6 |       23 |        3456 |       8 | Bob   |       |         | f         |        3
  7 |       89 |        2239 |       7 | Alice |       |         | f         |        2
(9 rows)
```

3)

Traditionally, Adding store_location id would lead to duplicates in much the same way that items already has for items of different prices.

Since Orders performs a join on items and retrieves prices for items, this propagates the redundancy to orders as well.

Proposed schematic view:



What I would do to make this solution more normalized is to introduce more atomicity into the data model. The current structure does not adhere to best practices when it comes to normalization.

This can be achieved by creating a Store table with ID which is connected to an item through Item ID and its associated pricing.

In our current situation, Store table would have Apple with Item ID 4 and 7 with respective prices across stores, say $2 and $2.5.

Later on, we can also condense item_id for apple into 1 and split it into more records based on price across different store_locations.

This would ease our pain that we are facing due to duplicates.

Part 2

```
CREATE TABLE product_categorizations ( id INTEGER NOT NULL CONSTRAINT
product_categorizations_pkey PRIMARY KEY, product_id INTEGER NOT NULL, category_id
INTEGER NOT NULL );
```

# Please note: Wrong name for constraint – foreign key cannot have the same name

1)

Short term – update values
Long term – DQ checks

The short term solution would be to perform update commands and do DQ checks in the long run to ensure correct mapping through a secondary validation pipeline.

However, this is a solution only for oranges being mapped incorrectly. The validation pipeline can work for numerous such cases but the update statements would pose an issue if this happens regularly for other items too.

In the interim, what we can do to mitigate this issue is to avoid taking the values from category key and instead load it from the product_categorization table since the correct mapping has been added by the analyst over there.

2)

If we are to add organic and regular, for example to our table, we will have to adopt the following strategy.

In the categorization table, we can add a column by the name of parent_id such that
Every category_id will have its parent id next to it. This will help us add as many mappings as possible and form a hierarchical mapping.

As shown in the image below, Produce is the highest category in the chain with null in parent id and the chain goes top-down to attach every category with its parent which also allows for great analytical capability with rolling up or drilling down.

Product

1	Organic orange
2	Reg orange

	Category	Parent id
1	Product	null
2	Fruits	1
3	Organic	2
4	Regular	2

	Product	Categ.
Prod.	1	1
Cat.	1	2
	1	3

In addition to this, the product categorization table also contains all products in addition to the categories they belong to introduce the concept without performing too many joins that provides for optimal querying.

3)

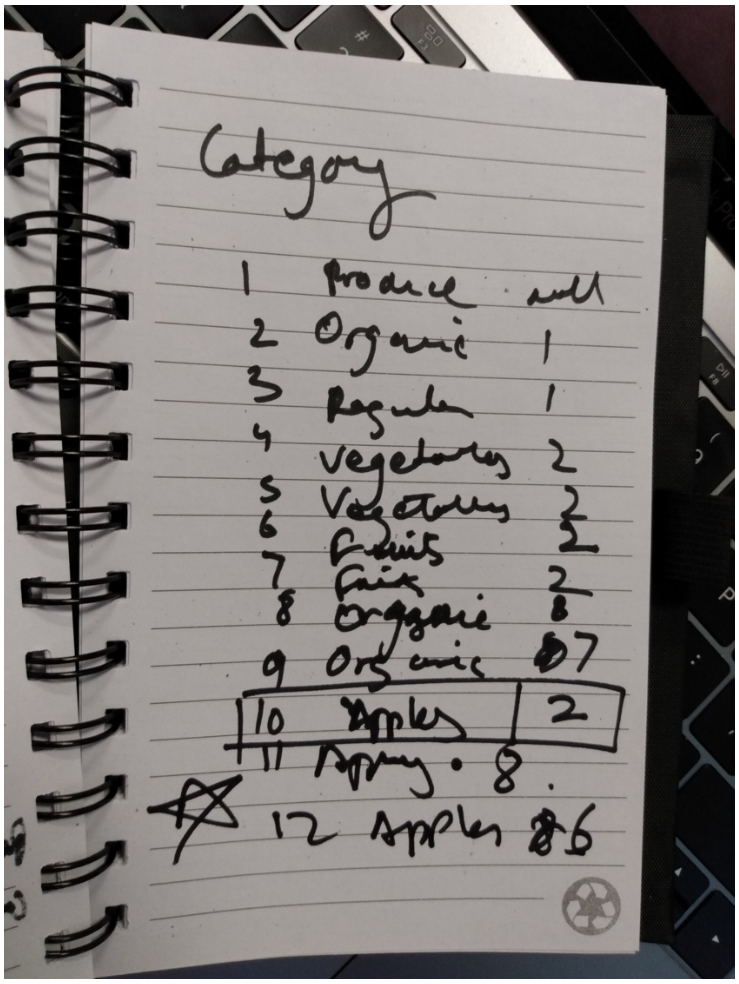
In the table: VALUES (9, 30275, 'Organic Apple', 'per lb', '2_PRODUCE_ORG_FRUITS');
Apple has a mapping but let us assume that it was null as follows
VALUES (9, 30275, 'Organic Apple', 'per lb', Null);

So how would we tackle this situation? I tried to incorporate this problem into the solution I built for the previous question.

As you may be able to see in the image, in the category table with parent_id introduced, we will be able to add apple to the most basic category automatically in the pipeline which is 'produce' which means that no product in the pipeline will be inserted without a category, thereby removing the risk of missing values in the database – therefore enforcing types. In this situation, we have set the default value to 'organic' for this specific situation since the scenario posed contained a solution for 'organic apples' but ideally it should be set as 'produce' as that is highest in the hierarchy.

An issue would be not being able to drill down for this particular product but we can add the capability in the pipeline to be able to distinguish a product to the next category be it fruit or vegetable or even, organic or regular. This would of course involve some parsing and pre-fed values – such as scraping articles being sold and then pre-assigning them values in a reference list as shown by the last two values in the image.

We can also manually insert specific categories if the products will null mappings are few which would be quick and a long-term fix at the same time.

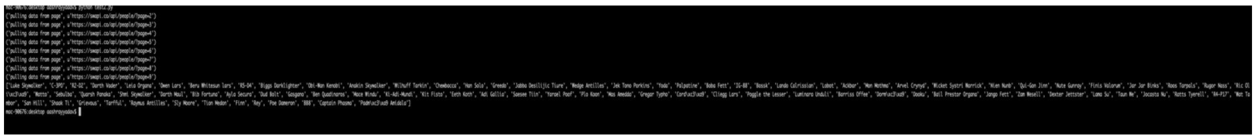

Category

1	Produce	null
2	Organic	1
3	Regular	1
4	Vegetables	2
5	Vegetables	2
6	Fruits	2
7	Fruits	2
8	Organic	8
9	Organic	~~8~~7
10	Apples	2
11	Apples	8
12	Apples	~~8~~6

Part 3

1)

List of all people



```
mac-90676:desktop aashrayyadav$ python test2.py
('pulling data from page', u'https://swapi.co/api/people/?page=2')
('pulling data from page', u'https://swapi.co/api/people/?page=3')
('pulling data from page', u'https://swapi.co/api/people/?page=4')
('pulling data from page', u'https://swapi.co/api/people/?page=5')
('pulling data from page', u'https://swapi.co/api/people/?page=6')
('pulling data from page', u'https://swapi.co/api/people/?page=7')
('pulling data from page', u'https://swapi.co/api/people/?page=8')
('pulling data from page', u'https://swapi.co/api/people/?page=9')
['Luke Skywalker', 'C-3PO', 'R2-D2', 'Darth Vader', 'Leia Organa', 'Owen Lars', 'Beru Whitesun lars', 'R5-D4', 'Biggs Darklighter', 'Obi-Wan Kenobi', 'Anakin Skywalker', 'Wilhuff Tarkin', 'Chewbacca', 'Han Solo', 'Greedo', 'Jabba Desilijic Tiure', 'Wedge Antilles', 'Jek Tono Porkins', 'Yoda', 'Palpatine', 'Boba Fett', 'IG-88', 'Bossk', 'Lando Calrissian', 'Lobot', 'Ackbar', 'Mon Mothma', 'Arvel Crynyd', 'Wicket Systri Warrick', 'Nien Nunb', 'Qui-Gon Jinn', 'Nute Gunray', 'Finis Valorum', 'Jar Jar Binks', 'Roos Tarpals', 'Rugor Nass', 'Ric Ol
i\xc3\xa9', 'Watto', 'Sebulba', 'Quarsh Panaka', 'Shmi Skywalker', 'Darth Maul', 'Bib Fortuna', 'Ayla Secura', 'Dud Bolt', 'Gasgano', 'Ben Quadinaros', 'Mace Windu', 'Ki-Adi-Mundi', 'Kit Fisto', 'Eeth Koth', 'Adi Gallia', 'Saesee Tiin', 'Yarael Poof', 'Plo Koon', 'Mas Amedda', 'Gregor Typho', 'Cord\xc3\xa9', 'Cliegg Lars', 'Poggle the Lesser', 'Luminara Unduli', 'Barriss Offee', 'Dorm\xc3\xa9', 'Dooku', 'Bail Prestor Organa', 'Jango Fett', 'Zam Wesell', 'Dexter Jettster', 'Lama Su', 'Taun We', 'Jocasta Nu', 'Ratts Tyerell', 'R4-P17', 'Wat To
mbor', 'San Hill', 'Shaak Ti', 'Grievous', 'Tarfful', 'Raymus Antilles', 'Sly Moore', 'Tion Medon', 'Finn', 'Rey', 'Poe Dameron', 'BB8', 'Captain Phasma', 'Padm\xc3\xa9 Amidala']
mac-90676:desktop aashrayyadav$ 
```

I retrieved this first and put a screen shot here because I were to retrieve list of all people in part 2 anyway and so I thought this would be a one-time display so as to remove redundancy in output.
2)

```
('List of ships for every Person: /n', {'Lobot': [], 'Gregar Typho': 'Naboo fighter', 'Leia Organa': [], 'Sly Moore': [], 'Luminara Unduli': [], 'Lando Calrissian': 'Millennium Falcon', 'Dooku': [], 'Taun We': [], 'Tion Medon': [], 'Chewbacca': 'Imperial shuttle', 'Ayla Secura': [], 'Saesee Tiin': [], 'Plo Koon': 'Jedi starfighter', 'Dud Bolt': [], 'Jar Jar Binks': [], 'Anakin Skywalker': 'Naboo fighter', 'Adi Gallia': [], 'Nute Gunray': [], 'Jango Fett': [], 'Bail Prestor Organa': [], 'Sebulba': [], 'Finn': [], 'Obi-Wan Kenobi': 'Belbullab-22 starfighter', 'Han Solo': 'Imperial shuttle', 'Rugor Nass': [], 'R2-D2': [], 'Rey': [], 'Mas Amedda': [], 'Boba Fett': 'Slave 1', 'Dorm\xc3\xa9': [], 'Biggs Darklighter': 'X-wing', 'Qui-Gon Jinn': [], 'Grievous': 'Belbullab-22 starfighter', 'C-3PO': [], 'Jabba Desilijic Tiure': [], 'Quarsh Panaka': [], 'Mace Windu': [], 'R4-P17': [], 'Raymus Antilles': [], 'Jocasta Nu': [], 'Poggle the Lesser': [], 'Captain Phasma': [], 'R5-D4': [], 'Bib Fortuna': [], 'Padm\xc3\xa9 Amidala': 'Naboo fighter', 'Jek Tono Porkins': 'X-wing', 'Watto': [], 'Eeth Koth': [], 'Beru Whitesun lars': [], 'Gasgano': [], 'Nien Nunb': 'Millennium Falcon', 'Ki-Adi-Mundi': [], 'Yoda': [], 'Dexter Jettster': [], 'Wicket Systri Warrick': [], 'Roos Tarpals': [], 'San Hill': [], 'Bossk': [], 'Owen Lars': [], 'Ric Oli\xc3\xa9': 'Naboo Royal Starship', 'Poe Dameron': 'T-70 X-wing fighter', 'Wat Tambor': [], 'BB8': [], 'Greedo': [], 'Ratts Tyerell': [], 'Zam Wesell': [], 'Darth Maul': 'Scimitar', 'Darth Vader': 'TIE Advanced x1', 'Cord\xc3\xa9': [], 'Tarfful': [], 'Barriss Offee': [], 'Finis Valorum': [], 'IG-88': [], 'Ben Quadinaros': [], 'Cliegg Lars': [], 'Arvel Crynyd': 'A-wing', 'Lama Su': [], 'Kit Fisto': [], 'Palpatine': [], 'Luke Skywalker': 'Imperial shuttle', 'Yarael Poof': [], 'Ackbar': [], 'Wilhuff Tarkin': [], 'Wedge Antilles': 'X-wing', 'Mon Mothma': [], 'Shaak Ti': [], 'Shmi Skywalker': []})
```

```
('List of planets for every Person: /n', {'Lobot': 'Bespin', 'Gregar Typho': 'Naboo', 'Leia Organa': 'Alderaan', 'Sly Moore': 'Umbara', 'Luminara Unduli': 'Mirial', 'Lando Calrissian': 'Socorro', 'Dooku': 'Serenno', 'Taun We': 'Kamino', 'Tion Medon': 'Utapau', 'Chewbacca': 'Kashyyyk', 'Ayla Secura': 'Ryloth', 'Saesee Tiin': 'Iktotch', 'Plo Koon': 'Dorin', 'Dud Bolt': 'Vulpter', 'Jar Jar Binks': 'Naboo', 'Anakin Skywalker': 'Tatooine', 'Adi Gallia': 'Coruscant', 'Nute Gunray': 'Cato Neimoidia', 'Jango Fett': 'Concord Dawn', 'Bail Prestor Organa': 'Alderaan', 'Sebulba': 'Malastare', 'Finn': 'unknown', 'Obi-Wan Kenobi': 'Stewjon', 'Han Solo': 'Corellia', 'Rugor Nass': 'Naboo', 'R2-D2': 'Naboo', 'Rey': 'unknown', 'Mas Amedda': 'Champala', 'Boba Fett': 'Kamino', 'Dorm\xc3\xa9': 'Naboo', 'Biggs Darklighter': 'Tatooine', 'Qui-Gon Jinn': 'unknown', 'Grievous': 'Kalee', 'C-3PO': 'Tatooine', 'Jabba Desilijic Tiure': 'Nal Hutta', 'Quarsh Panaka': 'Naboo', 'Mace Windu': 'Haruun Kal', 'R4-P17': 'unknown', 'Raymus Antilles': 'Alderaan', 'Jocasta Nu': 'Coruscant', 'Poggle the Lesser': 'Geonosis', 'Captain Phasma': 'unknown', 'R5-D4': 'Tatooine', 'Bib Fortuna': 'Ryloth', 'Padm\xc3\xa9 Amidala': 'Naboo', 'Jek Tono Porkins': 'Bestine IV', 'Watto': 'Toydaria', 'Eeth Koth': 'Iridonia', 'Beru Whitesun lars': 'Tatooine', 'Gasgano': 'Troiken', 'Nien Nunb': 'Sullust', 'Ki-Adi-Mundi': 'Cerea', 'Yoda': 'unknown', 'Dexter Jettster': 'Ojom',
'Wicket Systri Warrick': 'Endor', 'Roos Tarpals': 'Naboo', 'San Hill': 'Muunilinst', 'Bossk':
'Trandosha', 'Owen Lars': 'Tatooine', 'Ric Oli\xc3\xa9': 'Naboo', 'Poe Dameron': 'unknown', 'Wat
Tambor': 'Skako', 'BB8': 'unknown', 'Greedo': 'Rodia', 'Ratts Tyerell': 'Aleen Minor', 'Zam Wesell':
'Zolan', 'Darth Maul': 'Dathomir', 'Darth Vader': 'Tatooine', 'Cord\xc3\xa9': 'Naboo', 'Tarfful':
'Kashyyyk', 'Barriss Offee': 'Mirial', 'Finis Valorum': 'Coruscant', 'IG-88': 'unknown', 'Ben
Quadinaros': 'Tund', 'Cliegg Lars': 'Tatooine', 'Arvel Crynyd': 'unknown', 'Lama Su': 'Kamino', 'Kit
Fisto': 'Glee Anselm', 'Palpatine': 'Naboo', 'Luke Skywalker': 'Tatooine', 'Yarael Poof': 'Quermia',
'Ackbar': 'Mon Cala', 'Wilhuff Tarkin': 'Eriadu', 'Wedge Antilles': 'Corellia', 'Mon Mothma':
'Chandrila', 'Shaak Ti': 'Shili', 'Shmi Skywalker': 'Tatooine'})
```

3) For this part, the ideal way to find values would have been to take a nested approach i.e. go from planets[residents] -> people -> species and then reverse the dictionary.

While going through the API, I found that most of the species were linked to one homeworld. So, I created a simple dict and if any species had NULL for homeworld, it meant that they existed on more than one planet.

I have kept the code very simple and well commented for it to be easily interpretable. I have also used abstractions to get list of ships and planets since it could be abstracted.

You may comment out the last function if you do not wish to call it and only run part 1 and 2.