Group 5 (Aidan Webster, Karli Young, Nop Lertsumitkul)

CS3200 Peer Review for Group 4 (AMC Records)

Clarity:

- For TicketType, are the current type inputs “3D”, “Normal”, and “Matinee”? Specifically defining all possible inputs would be useful in the description to better understand what factors into price.
- Is Movie.rating equivalent to Ticket_Type.age_range? This seems to be what the description implies but the two are never linked. It would likely be too disconnected if it were tracked separately in Ticket_type like in ERD.

Organization:

- Logically organized, a minor mislabeling of queries 1 and 3 under “Transactions”.

Depth:

- In general, the database is really useful for looking at broad trends and information after the fact, but is a little less straightforward for planning more day-to-day logistical things a movie theater might need. For example:
    - Since a whole history (ie. past, present and in the case of movies, future) of both employees and movies are being kept, picking out only currently employed employees (perhaps for scheduling purposes) or movies that are either historical, now playing or upcoming, has to happen with a date comparison or null check instead of explicit attribute check. For employees, it may make sense to have a derived attribute that checks if they’re currently employed. For movies, a release and possibly end date attribute may be useful instead of relying on the connection to screenings for that info (if screenings are created a month in advance, then query 1 would not return upcoming movies that begin screening two weeks from now, since it will now have scheduled screenings in the system, just ones that haven’t happened yet).
- It would make sense for tickets to also collect information on which particular seat is booked. If the seats that have been booked are known, this helps customers plan bookings, as well helps theatres control where customers can sit. Some theatres have booking rules that do not allow 1 space to be left in between bookings (since people don’t want to sit directly next to strangers). If a customer complained about their experience, the ticket showing the seat can help management determine if maybe it was not cleaned properly.

Breadth:

- Some potential additions to the schema could include an open_date for each theatre and customer reward points. An open_date attribute would allow managers to compare how well a theatre is performing given how new or old it is. Marketing can then allocate resources to those theatres as well as prepare in advance for new theatre openings.
- Some theatres might not have dedicated parking lots, like those in shopping malls. Adding another table for parking lots would be useful. This is more prevalent in other countries, which would make this database model applicable internationally. The field for parking lot in the Theatre table should be a foreign key to the parking lot table, which can also be null. Also, is showing parking even necessary? There aren’t any meaningful queries involving parking lots. So would this just overcomplicate the database?

SQL:

- There are some minor issues with additional spaces in the txt file when running in pgAdmin.
- Query 4 returns all ids, it might be more useful for the business users if the query returned the names of movies and customers instead of arbitrary id numbers that need to be further decoded.

- Since it tracks past employees, the only indicator of whether or not they currently work at a particular theater is if the current date falls within the start and end date (or if the end date is null??). For scheduling employees to work (or anything dealing with current employees only), it adds an extra step to make sure past employees are excluded. Maybe adding a derived attribute from start/end dates that says if they're currently employed
- A release and end date or some other kind of attribute might be useful for a movie to determine if it's now playing in theaters. Right now, finding past, present or future movies relies heavily on screenings, which are arbitrary
- Movie is missing age_range in ERD (but the description says it would track it in Movie… it would be a little too disconnected if it were only tracked in Ticket_type like in ERD)
    - It's also worth considering how and when screenings are made, because query 1, which finds upcoming movies, would change if screenings were made ahead of time for upcoming movies

Add an open_date for each theatre. This would allow managers to compare how well a theatre is performing given how new or old it is. Marketing can then allocate resources to those theatres as well as prepare in advance for new theatre openings.

Tickets should also collect information on which particular seat it is for. If the seats that have been booked are known, this helps customers plan bookings, as well as theatres control where customers can sit. Some theatres have booking rules that do not allow 1 space to be left in between bookings (since people don't want to sit directly next to strangers). If a customer complained about their experience, the ticket showing the seat can help management determine if maybe it was not cleaned properly.

Some theatres might not have dedicated parking lots, like those in shopping malls. Adding another table for parking lots would be useful. This is more prevalent in other countries, which would make this database model applicable internationally. The field for parking lot in the Theatre table should be a foreign key to the parking lot table, which can also be null. Also, is showing parking even necessary? There aren't any meaningful queries involving parking lots. So would this just overcomplicate the database?

Group 5 (Aidan Webster, Karli Young, Nop Lertsumitkul)

CS3200 Project - Phase 2

For our project, we chose to study a local cafe chain (something like Tatte).

Business users and their expectations from the database:

Firm HR department needs to track employees across all locations and utilize the database in order to maintain up-to-date information about employee contact information, pay rate, and primary store location. HR would also want the database to collect sale and shrinkage data to perform analysis on employee performance.

Firm financial department needs to collect information about the trend of inventory usage and purchases from suppliers to estimate how much the company should expect to spend on raw materials in the future.

Cafe chain executives would rely on the database to collect information to make strategic decisions for the firm with a holistic view of the data. One example would be tracking item popularity using point-of-sale data from the database to consider menu changes to add/remove items, make alterations to prices, or offer seasonal goods. Information about each store's performance would allow executives to identify locations for future stores, make decisions about which stores to close, and compare shrinkage levels between stores. Additionally, executives would need inventory information to adjust supplier order levels to ensure each store is adequately prepared for customer demand on any given day/week/season without over-ordering (note: we assume store managers only keep the information such as inventory levels and delivery completion updated, but higher executives are the end users that ultimately rely on/use the data for business purposes).

How our solution supports or enhances new or existing business processes:

1. Inventory management

The company will have to record each piece of inventory that is used in production. Comparing the monthly inventory to sales can help managers to identify which products are using the most inventory. Furthermore, if some inventory or sales are not recorded, the difference between these two records will notify managers about the mistake, which can be corrected accordingly. Losses in inventory can also be attributed to human error during the production of food/drink, which can be acted upon to encourage employees to efficiently use the inventory. Keeping track of inventory depletion rates can also help management plan the next order of raw materials depending on the current usage.

2. Comparing online vs in-person ordering

Online ordering has become immensely popular in recent years and firms are having to decide how much of the firm's resources to dedicate to further developing online capabilities vs in-person functionalities. Using this database, store executives can compare rates of in-person vs online ordering, and perform analysis on which type of order is more likely to be associated

with a customer account, which type of order is likely to be larger, and which type of order is more expensive for the firm to process (credit vs cash purchases).

3. How does store performance compare across the chain?

Executives will want to compare the total costs/revenues and the performance of various products across different store locations in order to inform decisions about budget distribution, opening/closing locations, and restocking inventory (our assumption is that inventory going from suppliers to stores is planned by a subscription model where executives decide the frequency and amount of inventory going to the stores on a daily or weekly basis, and they only need to adjust their inventory order when they want to). For example, they may need to allot more of the budget to a store with much more business, in order for them to keep up or they may decide to close one location that is doing significantly worse than all the other locations.

Database requirements, including regular updates/modifications needed:

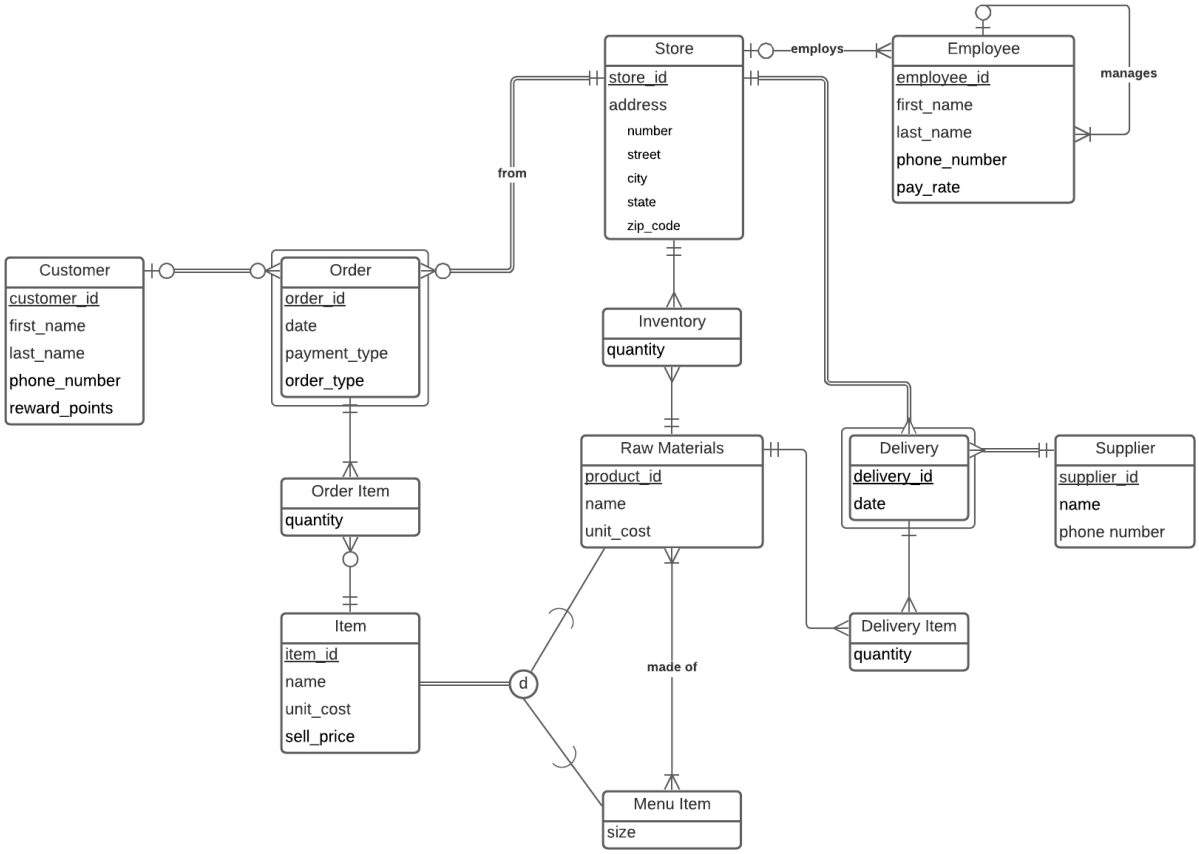
For our project, we are tracking a chain of coffeehouse stores, which are at different locations. For each store, we will track who its employees are and who the managers are. Stores must employ at least one employee. Managers must manage at least one employee to be a manager, but some employees may not have a manager (for example, if they are the highest manager). Some employees are not associated with stores because they are higher level management that don't work with any one particular store.

Each store will keep track of its inventory of raw materials. Raw materials are the items delivered directly to the store. A raw material provided by a specific supplier is uniquely identified by its product ID. For example, milk supplied by Clover farms is distinct from milk supplied by Darigold, as the unit costs and quantity possessed by the store of each may differ. The raw materials get to the store via deliveries, which occur at regular intervals (ie. weekly). Raw materials may either be sold as is (for example, bottled water, or other prepackaged items), or will be used as ingredients to make a menu item (for example, 6 oz milk, 3 tsp. sugar, 8 oz coffee grinds and 10 oz hot water make a coffee). We assume menu items are all drinks, so it has a size associated with it. Prepackaged items and menu items are both types of Items that may be purchased by customers, so they have a sell price associated with them. They also have a unit cost, which is the same as the raw material unit cost for prepackaged items and the cost of producing a drink (ie. combination of ingredients at various quantities) for the menu item.

We will also be tracking every order at each store. An order goes to exactly one store, may contain any amount of items (different or the same), may be bought online or in-store, is bought with some kind of payment type, and may either be bought by a loyalty customer whose information we have on file (so that we can give them points for their purchases), or an anonymous guest customer. For example, we would be able to model someone purchasing 1 bottle of orange juice, 8 mocha lattes, or someone purchasing both 1 bottle of orange juice and 8 mocha lattes at once.

Some regular updates and modifications are needed to maintain the database. Regular updates include Order entries after each transaction, Raw Materials entries after each delivery, and Raw Materials update at store close every day (to monitor shrinkage). Any other additions to Supplier, Employee, Store, Item, etc. should be updated as needed.

E/R Diagram:



https://lucid.app/lucidchart/invitations/accept/e143d3f6-3dd0-47d9-929f-8074206e7d53

Group 5 (Aidan Webster, Karli Young, Nop Lertsumitkul)

CS3200 Project - Phase 3

Summary and problem context

For our project, we are examining a local coffeehouse chain. The chain's executives, financial department, and HR department need records of various data in order to ensure a smooth operation of the business. Our solution is a database that keeps track of all the stores in the chain, the employees that work there, customers that shop there, and the inventory flow in and out of each store. With all this information, there are many useful outcomes of the database; we have identified three in particular below.

Benefits/objectives of the database

1. Inventory management

The company will have to record each piece of inventory that is used in production. Comparing the monthly inventory to sales can help managers to identify which products are using the most inventory. Furthermore, if some inventory or sales are not recorded, the difference between these two records will notify managers about the mistake, which can be corrected accordingly. Losses in inventory can also be attributed to human error during the production of food/drink, which can be acted upon to encourage employees to efficiently use the inventory. Keeping track of inventory depletion rates can also help management plan the next order of raw materials depending on the current usage.

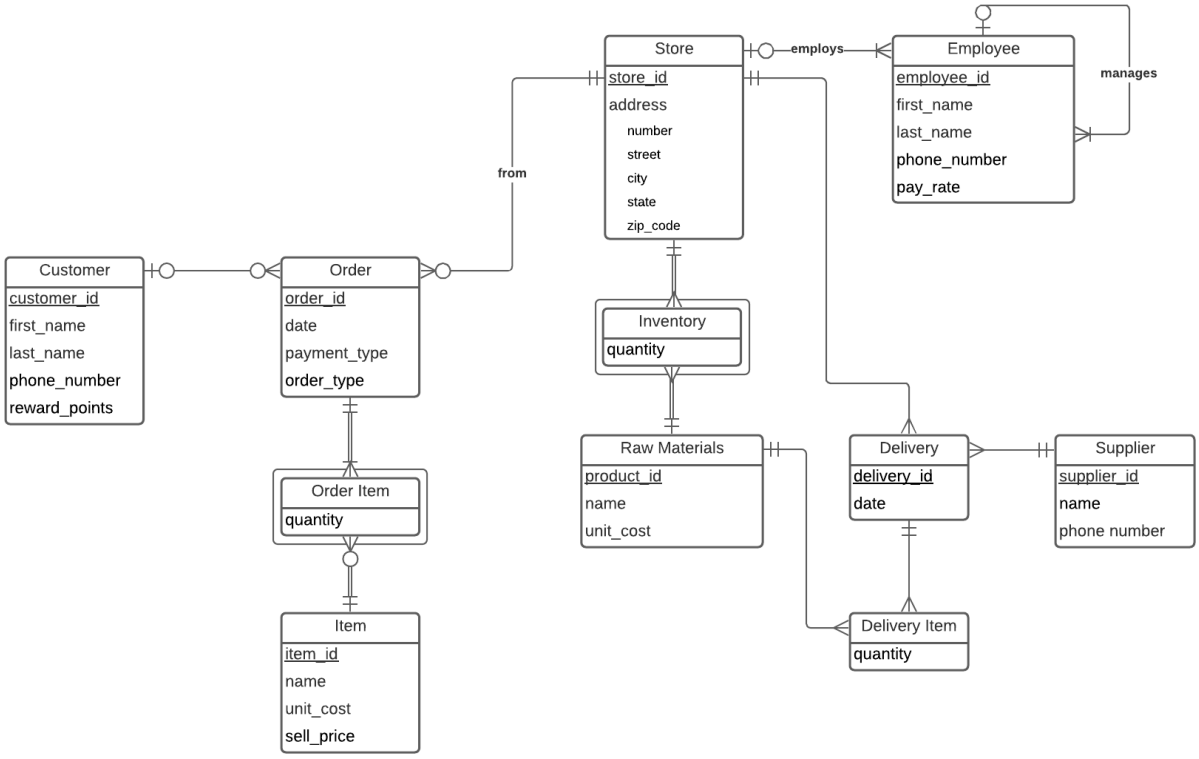
2. Comparing online vs in-person ordering

Online ordering has become immensely popular in recent years and firms are having to decide how much of the firm's resources to dedicate to further developing online capabilities vs in-person functionalities. Using this database, store executives can compare rates of in-person vs online ordering, and perform analysis on which type of order is more likely to be associated with a customer account, which type of order is likely to be larger, and which type of order is more expensive for the firm to process (credit vs cash purchases).

3. How does store performance compare across the chain?

Executives will want to compare the total costs/revenues and the performance of various products across different store locations in order to inform decisions about budget distribution, opening/closing locations, and restocking inventory (our assumption is that inventory going from suppliers to stores is planned by a subscription model where executives decide the frequency and amount of inventory going to the stores on a daily or weekly basis, and they only need to adjust their inventory order when they want to). For example, they may need to allot more of the budget to a store with much more business, in order for them to keep up or they may decide to close one location that is doing significantly worse than all the other locations.

Entity-Relationship Diagram

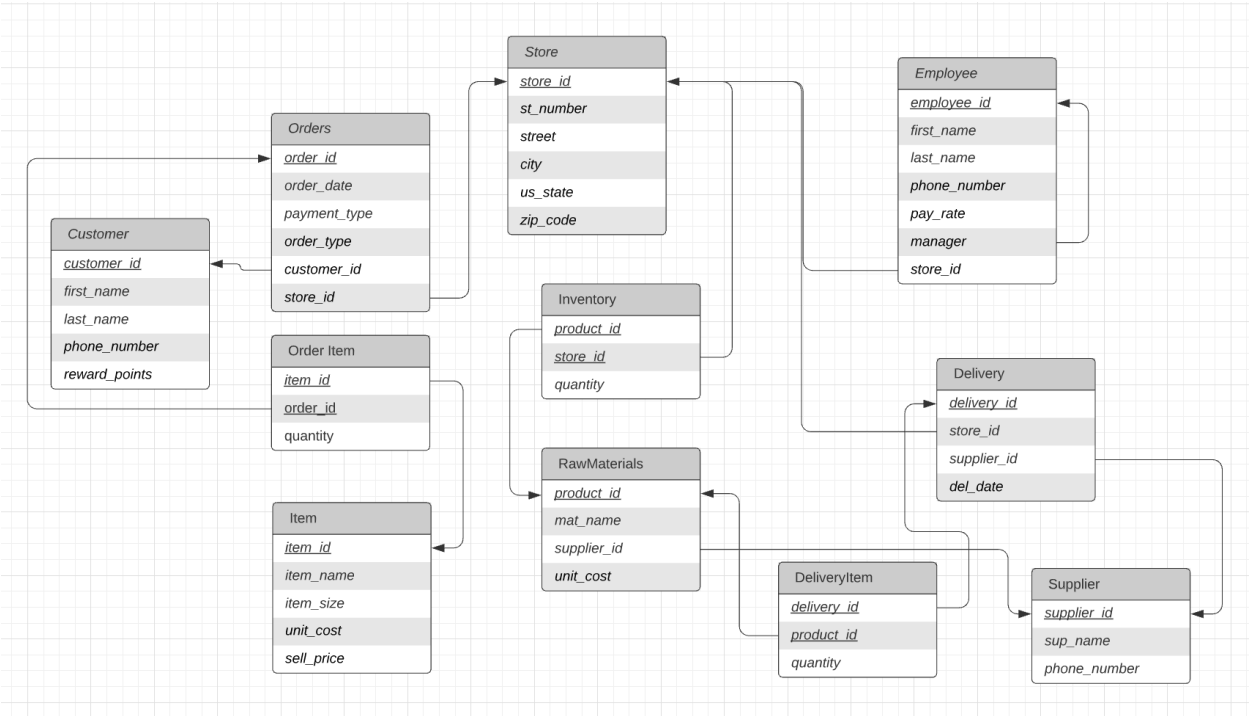


With this ER diagram, we are able to determine a wide range of relationships within a cafe chain. For example, we need to model the employee structure, customer orders, and inventory management. These relationships are centered around the store entity in this ER diagram.

A high-level description of the model: There are multiple stores in the cafe chain. Stores employ employees, some of which are managers. Each store has an inventory of raw materials (including both *ingredients* for menu items like milk, coffee beans, etc. and *prepackaged items* like bottled water). Raw materials are delivered in various quantities to each store by suppliers. We keep track of all the orders of each store, and the items sold through the order. These items might be something off the menu, like a cup of coffee, or it may be a prepackaged item (which is a raw material since it is delivered from the supplier as is, as was mentioned earlier). Some customers are kept in the system for loyalty points, and we match them to their orders if they are represented in the system.

With the way we have modelled this structure, the entities for raw materials, items, and customers are not tied to a particular store. Therefore, the entries for these tables only need to exist once, and can be referred to through relationships like being in an order at a particular store, or inventory. This reduces the amount of unnecessary data stored.

Relational Schema
Normalized 3NF:



This relational schema will help to illustrate the practical usage of the database to obtain certain information. For example, if a customer was served spoiled milk, we would be able to join tables from Customer all the way to Supplier to determine which supplier and delivery was responsible. This would allow management to alert all stores with deliveries from the same supplier on the same date of the issue, and have their inventories checked.

SQL queries that implement previous business processes

1. Inventory management
    a. Scenario 1a: Executive in charge of managing inventory levels wants to check how much of each material each store has at a given point in time. (The query checks inventory levels for store #3, but a user could theoretically input whichever store number they want to view).

```
SELECT rm.product_id, i.quantity, rm.mat_name, rm.unit_cost
FROM Inventory i
JOIN RawMaterials rm ON rm.product_id = i.product_id
WHERE store_id = 3
```

    b. Scenario 2b: Executives want to know which product has the highest in-stock inventory across all stores, as an indicator of underperformance. *(Witness query, Having clause)*

```
SELECT rm.product_id, rm.mat_name, sum(quantity)
FROM RawMaterials rm
JOIN Inventory i on i.product_id = rm.product_id
GROUP BY rm.product_id
HAVING sum(quantity) >= ALL ( SELECT sum(quantity)
                              FROM RawMaterials rm
                              JOIN Inventory i
                              on i.product_id = rm.product_id
                              GROUP BY rm.product_id )
```

2. Comparing online vs in-person ordering
    a. Online ordering has become immensely popular in recent years and firms are having to decide how much of the firm's resources to dedicate to further developing online capabilities vs in-person functionalities. Using this database, store executives can compare rates of in-person vs online ordering, and perform analysis on which type of order is more likely to be associated with a customer account. Here marketing teams can find how many users have chosen not to provide personal information in-person and online.

```
Old:
--Of the sample data, which order type is associated with anonymous
--customers (no customer data is maintained for reward_points)
SELECT o.order_type, count(c.customer_id)
Number_of_Anonymous_Customers
FROM Orders o
JOIN Customer c on o.customer_id = c.customer_id
JOIN OrderItem oi on o.order_id = oi.order_id
GROUP BY c.customer_id, o.order_type
HAVING c.customer_id in
       (SELECT c.customer_id as non_null
       FROM Orders o
       JOIN Customer c on o.customer_id = c.customer_id
       JOIN OrderItem oi on o.order_id = oi.order_id
       WHERE c.reward_points is Null)

Revised:
--Of the sample data, which order type is associated with anonymous
--customers (no customer data is maintained for reward_points)
SELECT o.order_type, count(c.customer_id)
Number_of_Anonymous_Customers
FROM Orders o
JOIN Customer c on o.customer_id = c.customer_id
JOIN OrderItem oi on o.order_id = oi.order_id
JOIN (SELECT c.customer_id as non_null
```

```
        FROM Orders o
        JOIN Customer c on o.customer_id = c.customer_id
        JOIN OrderItem oi on o.order_id = oi.order_id
        WHERE c.reward_points is Null) as X
ON c.customer_id = X.non_null
WHERE X.non_null is not Null
GROUP BY o.order_type
```

b. Scenario 2b: Executives want to understand which order type (in-person vs online) accounts for higher revenue and COGS.

```
Old:
--Of the sample data, how much revenue was generated by in person
--sales and online sales, and what was the cost of goods sold for each
SELECT o.order_type, sum(i.unit_cost) cogs, sum(i.sell_price) revenue
FROM Orders o
JOIN Customer c on o.customer_id = c.customer_id
JOIN OrderItem oi on o.order_id = oi.order_id
JOIN Item i on oi.item_id = i.item_id
GROUP BY o.order_type

Revised:
--Of the sample data, how much revenue was generated by in person
--sales and online sales, and what was the cost of goods sold for each
SELECT o.order_type, sum(i.unit_cost*oi.quantity) cogs,
sum(i.sell_price*oi.quantity) revenue
FROM Orders o
JOIN OrderItem oi on o.order_id = oi.order_id
JOIN Item i on oi.item_id = i.item_id
GROUP BY o.order_type
```

c. Scenario 2c: The marketing department wants to know which customers have only utilized online ordering. *(Outer join)*

```
SELECT c.customer_id, c.first_name, c.last_name
FROM Orders o
JOIN Customer c on o.customer_id = c.customer_id
LEFT JOIN
        (SELECT c.customer_id cust_id
        FROM Orders o
        JOIN Customer c on o.customer_id = c.customer_id
        WHERE o.order_type = 'in-person') as X
        ON c.customer_id = X.cust_id
WHERE X.cust_id is Null
GROUP BY c.customer_id
```

3. How does store performance compare across the chain?
    a. Scenario 3a: Executives want to know how many sales were made and how many customers shop at each store. They want to be able to view the stores in order of their performance with regard to the number of sales made.

```
SELECT s.store_id, count(distinct o.order_id) numsales,
       count (distinct o.customer_id) as numcustomers
FROM Store s
JOIN Orders o on s.store_id = o.store_id
JOIN OrderItem oi on oi.order_id = o.order_id
JOIN Item i on oi.item_id = i.item_id
GROUP BY s.store_id
ORDER BY numsales desc
```

    b. Scenario 3b: Executives want to know what each store's gross revenue, cost of goods sold and net revenue was for the month of January.

```
SELECT s.store_id,
       sum(oi.quantity*i.sell_price) as gross_rev,
       sum(oi.quantity*i.unit_cost) as cogs,
       sum(oi.quantity*i.sell_price - oi.quantity*i.unit_cost) as
net_rev
FROM Store s
JOIN Orders o on s.store_id = o.store_id
JOIN OrderItem oi on oi.order_id = o.order_id
JOIN Item i on oi.item_id = i.item_id
WHERE o.order_date > '2019-12-31'
AND o.order_date < '2020-02-01'
GROUP BY s.store_id
ORDER BY s.store_id
```

Routine Updates:

1) For each point-of-sale transaction, a new tuple must be added to the database to reflect the sale. Below is an example tuple where order # is 1 + the last order number used, order data is Dec 4 of 2020, the payment type is debit, the order type in in-person, the customer who made the purchase is stored in the customer table using the id 00046313, and the transaction was at store 001.

```
INSERT INTO Orders (order_id, order_date, payment_type, order_type,
customer_id, store_id)
VALUES
(00046313, '2020-12-04', 'debit', 'in-person', 002354, 001);
```

2) Inventory quantity needs to be updated regularly at the end of every day to reflect actual stock levels for each product at each store.

```
UPDATE Inventory
SET quantity = 50
WHERE product_id = 021 and store_id = 003;
```

What we learned

This project turned out to be a lot more nuanced than we initially thought. We found that having well thought-out real world requirements from the beginning was vitally important. These requirements set the model for the entire project, so having clear requirements led to a clearer conceptual model, while ambiguous guidelines led to ambiguity in the model. Plus, finding an inconsistency or mistake in the conceptual model as a result of ambiguity later on makes the whole thing much harder to fix.

We were also challenged throughout this project to consider the practical uses/benefits and usage of a database. This helped us to make certain design decisions, like which attributes were necessary or trivial, which information is relevant to other information (to determine foreign keys), cardinalities of relationships, etc. We sometimes had to make certain assumptions, refer back to the scope often, and have discussions about benefits vs. drawbacks to inform these decisions.

Project work distribution

We each took a little bit of a lead on different aspects of the project based on what we felt comfortable with (Karli on ERD, Nop on relational schema reduction and normalization, Aidan on SQL implementation and business usages/requirements), but ultimately worked together on everything. We met with each other often and kept in constant communication to make sure we were all kept updated about the progress of the project.

Example Tuples:

Customer				
customer_id	**first_name**	**last_name**	**phone_number**	**reward_points**
010017	John	Rose	617-555-3646	0
002302	Jane	Baker	415-555-0413	75
000118	Null	Null	Null	Null
009640	Aubrey	Morgan	617-555-1172	25
000546	Max	Alexander	718-555-8755	1050

Orders					
order_id	**order_date**	**payment_type**	**order_type**	**customer_id**	**store_id**
00020312	2019-12-29	debit	in-person	009640	003
00542801	2020-01-02	credit	online	000546	004
00012438	2020-01-08	cash	in-person	002302	002
00445390	2020-01-12	credit	online	000546	004
00037577	2020-01-15	cash	in-person	000118	001

OrderItem		
quantity	**order_id**	**item_id**
2	00020312	002
1	00542801	028
2	00012438	005
1	00445390	001
8	00037577	001
2	00542801	006
1	00445390	004

Item				
item_id	**item_name**	**item_size**	**unit_cost**	**sell_price**
001	coffee	small	$0.75	$1.50
002	latte	small	$1.50	$3.00
003	mocha	small	$1.75	$3.50
004	hot chocolate	small	$0.75	$2.50
005	chai tea	small	$1.25	$3.00
006	coffee	large	$1.00	$1.75
007	latte	large	$1.75	$3.50
008	mocha	large	$2.00	$4.00
009	hot chocolate	large	$1.00	$3.00
010	chai tea	large	$1.50	$4.00
028	water bottle	Null	$0.10	$1.00

Inventory		
product_id	**store_id**	**quantity**
021	001	10
022	001	23
021	002	7
025	002	11
021	003	6
022	003	50
021	004	4
022	004	26
023	004	13
028	004	112

RawMaterials			
product_id	**mat_name**	**supplier_id**	**unit_cost**
021	whole milk	001	$4.00
022	coffee grounds	005	$7.00
023	cacao	004	$2.00
024	half-and-half	001	$2.00
025	chai tea	006	$2.50
026	sugar	003	$1.50
027	whipped cream	001	$1.50
028	water bottle	002	$0.10

Delivery			
delivery_id	**store_id**	**supplier_id**	**del_date**
0046	001	001	2019-12-20
0110	002	006	2019-12-23
0356	003	005	2019-12-26
0212	004	002	2019-12-28

DeliveryItem		
delivery_id	**product_id**	**quantity**
0046	021	10
0110	025	200
0356	022	50
0212	028	240

Supplier		
supplier_id	**sup_name**	**phone_number**
001	Clover	617-555-4377
002	Dasani	617-555-0365
003	Domino	617-555-1234
004	Hershey's	617-555-4529
005	Equator	617-555-9093
006	Twinning's	617-555-9376

Store					
store_id	**st_number**	**street**	**city**	**us_state**	**zip_code**
001	04	Quincy Ave.	Boston	MA	02115
002	112	Main St.	Boston	MA	02114
003	4746	Washington Rd,	Cambridge	MA	02115
004	220	Huntington Ave.	Boston	MA	02120

Employee						
employee_id	**first_name**	**last_name**	**phone_number**	**pay_rate**	**manager**	**store_id**
0016	Jennifer	Johnson	617-555-9900	$35	Null	Null
0101	Michael	Andrews	617-555-7777	$12	0101	002
0047	Kimberly	Paul	617-555-4448	$18	0016	002
0099	Michelle	Lowe	617-555-8688	$18	0016	004
0112	Charles	Robertson	718-555-4321	$12	0099	004