MySQL journal for Mexico toy sales

Cleaning up Products Table

- Due to columns `Product_Cost` and `Product_Price` both have $ signs, it means that they are both texts, instead of being floats.

```
-- Remove ($) signs from the data, since SQL won't be able to calculate.
UPDATE toy_sales.products
SET `Product_Cost` = REPLACE(`Product_Cost`, '$',''),
 `Product_Price` = REPLACE(`Product_Price`, '$','');
```

1. Which product categories drive the biggest profits? Is this the same across store locations?
- Im going to use JOIN for `products` (since it shows how many units is sold), `sales`` (shows prices) and `stores` (shows stores and store location)

Table	Primary key	Foreign key
Products	Product ID	
Sales	Sales ID	Product id, store id
stores	Store id	

- The columns I'm selecting are:
    - `Product_name and product_category` - Since I want to have the name of the product name and the category, since the question wants to know the result of what category generates the most profit. I feel like knowing the name would definitely help see the profit generated in different products in the same category
    - Product_price, product_cost, COUNT(units) as total_counts - so we can see the profit. I use count to count all of the units, but since they are going to be grouped together, I can see the amount of units that are being sold by each product.

- Store_name and store_location - we can see the stores and where they are, since there may be an affect of the location.

```sql
-- joins tables products, stores, and sales and put them together as one whole table

WITH cte AS (
SELECT pr.`Product_Name`, pr.`Product_Category`, pr.`Product_Cost`, pr.`Product_Price`,COUNT(sa.`Units`) AS total_units,
 st.`Store_Name`, st.`Store_Location`
FROM toy_sales.sales AS sa
INNER JOIN
toy_sales.products AS pr
ON sa.`Product_ID` = pr. `Product_ID`
INNER JOIN
toy_sales.stores AS st
ON sa.`Store_ID` = st.`Store_ID`
GROUP BY pr.`Product_Name`, pr.`Product_Category`, pr.`Product_Cost`, pr.`Product_Price`, st.`Store_Name`, st.`Store_Location`
ORDER BY `Product_Name` ASC
)

SELECT *, ROUND((total_units * `Product_Price`) - (total_units * `Product_Cost`), 2) AS `profit`
FROM cte; -- generates table with profit
```

I'm going to create this as a view as join_one, because this is going to be used as a table to answer two parts of the question.

1. Which product categories generate the most product
2. Is it the same across all stores????

Which product categories generate the most product?

```sql
-- Gets the top ten categories
SELECT
    `Product_Category` , `profit`
FROM
    toy_sales.join_one
GROUP BY  `Product_Category`, `profit`
ORDER BY `profit` DESC
LIMIT 10
```

A: Electronics are all in the top ten in profit, coming out as the most profitable category.

Is it the same across all store locations?

```sql
-- shows whether it's the same when store location is a factor
SELECT  `Store_Location`, `Product_Category`, `profit`
 FROM toy_sales.join_one
 GROUP BY `Store_Location`, `Product_Category`, `profit`
 ORDER BY `Store_location`,`profit` DESC;
```

A: Throughout all of the store locations, Electronics was the number one profitable category. Every location had either games or toys as second most profitable category.

2. Can you find any seasonal trends or patterns in the sales data?

- We have to find the date of when the products were being sold
- The profit in those seasons
- Need products and sales tables (JOIN)

```sql
 -- a cte that puts it in seasons
 WITH `cte` AS (
SELECT *, CASE
WHEN MONTH(`date`) IN (12,1,2,3,4) THEN 'DRIEST'
WHEN MONTH(`date`) IN (7,8) THEN 'SHOULDER'
WHEN MONTH(`date`) IN (5,6,9,10,11) THEN 'LOW'
ELSE 'No month available'
END AS `seasons`
FROM toy_sales.sales
),

-- gets the prices of the products, products name, and category
 cte_1 AS (
SELECT pr.`Product_Name`, pr.`Product_Category`, pr.`Product_Cost`, pr.`Product_Price`,COUNT(sa.`Units`) AS `total_units`,
sa.`seasons`
FROM `cte` AS `sa`
INNER JOIN
toy_sales.products `pr`
ON sa.`Product_ID` = pr.`Product_ID`
GROUP BY `Product_Name`, `Product_Category`, `Product_Cost`, `Product_Price`, `seasons`)

-- gets the profit and displays top ten results.
SELECT *, ROUND((`total_units` * `Product_Price`) - (`total_units` * `Product_Cost`)) AS `profit`
FROM `cte_1`
ORDER BY `profit` DESC
LIMIT 10;
```

- This SQL query creates two CTE's and the first cte has a case statement that categorizes which a month to a season. I searched up that Mexico has three seasons in terms of weather. The second CTE collects the price of the products and categories. Also, we get the total units that was sold. The main query is the

one that calculates the profit. Also it gets the top ten profitable products with their seasons

- Driest. December - april (when most of the country is at its driest)
- Shoulder (between spring and summer) - July and august
- Low (dominated by rain season) - May, June, and september - November
- Source - https://www.travelandleisure.com/trip-ideas/best-time-to-visit-mexico

Product_Name	Product_Category	Product_Cost	Product_Price	total_units	seasons	profit
Colorbuds	Electronics	6.99	14.99	32933	DRIEST	263464
Colorbuds	Electronics	6.99	14.99	25890	LOW	207120
Action Figure	Toys	9.99	15.99	22218	DRIEST	133308
Colorbuds	Electronics	6.99	14.99	14165	SHOULDER	113320
Action Figure	Toys	9.99	15.99	18763	LOW	112578
Lego Bricks	Toys	34.99	39.99	20843	LOW	104215
Lego Bricks	Toys	34.99	39.99	20686	DRIEST	103430
Deck Of Cards	Games	3.99	6.99	31536	DRIEST	94608
Deck Of Cards	Games	3.99	6.99	23861	LOW	71583
Nerf Gun	Sports & Outdoors	14.99	19.99	11949	DRIEST	59745

A: Half of the list is filled with products that that profited the most during the driest seasons. However, ‘Low’ seasons are not that far off, with having 4 ‘low’ seasons being at the top ten. There’s only one product that had a huge profit in the Shoulder season.

- Since the DRIEST months consist of December - April, it can be assumed that the reason why the profit is really high around that time is because of the Christmas Holidays and Easter.

3. Are sales being lost with out-of-stock products at certain locations?
    - Im going to use JOIN with inventory, products, stores, and sales.

What columns am I going to use?

- Store ID, product ID, stock_on_hand (to know whats in the inventory and use the ids to use join)

- Product Name, cost, price
- Get the store name and their location
- Units (to see how much is sold) - we can use this to see how much is sold, but also if we can see how much could have been sold if there was any products that are not in the inventory.

```sql
CREATE VIEW toy_sales.`zero` AS
-- takes store, the product, and the amount of units in the inventory
SELECT st.`Store_ID`, st.`Store_Name`,st.`Store_Location`, pr.`Product_ID`, pr.`Product_Name`, i.`Stock_On_Hand`
FROM toy_sales.`inventory` AS `i`
INNER JOIN
toy_sales.`stores` AS `st`
ON i.`Store_ID` = st.`Store_ID`
INNER JOIN
toy_sales.`products` AS `pr`
ON i.`Product_ID` = pr.`Product_ID`
INNER JOIN
toy_sales.`sales` AS `sa`
ON i.`Store_ID` = sa.`Store_ID`
WHERE `Stock_ON_Hand` = 0
GROUP BY `Store_ID`,`Store_Name`, `Store_Location`, `Product_ID`, `Product_Name`, `Stock_On_Hand`
ORDER BY `Store_ID` ASC, `Product_ID` ASC;
```

- I created a view (‘zero’) that would get the store_id, store, product_id, product, and the stock on hand.
- I filtered stock_on_hand = 0 so I can find the stores that have 0 units in total of a certain product.
- I’m using this as info on what type of sales are being lost.

I’m going to create another query that gets those products that had the stock_on_hand = 0 and then find how many units are sold throughout Mexico. I’m going to create it as a CTE and I’m going to combine it with the view, ‘zero’.

```sql
WITH `cte` AS (
SELECT pr.`Product_ID`, pr.`Product_Name`, COUNT(sa.`Units`) AS `units_sold_in_Mexico`
FROM toy_sales.sales `sa`
INNER JOIN
toy_sales.products `pr`
ON sa.`Product_ID` = pr.`Product_ID`
WHERE sa.`Product_ID` IN (1,2,3,4,5,9,11,12,13,14,15,16,21,23,28,29,31,32,33,34)
GROUP BY  `Product_ID`, `Product_Name`)

SELECT z.`Store_Name`, z.`Store_Location`, z.`Product_Name`, z.`Stock_On_Hand`, `cte`.`units_sold_in_Mexico`
FROM toy_sales.zero `z`
INNER JOIN
`cte`
ON z.`Product_ID` = `cte`.`Product_ID`
GROUP BY `Store_Name`, `Store_Location`, `Product_Name`, `Stock_On_Hand`, `units_sold_in_mexico`
ORDER BY `Store_location` ASC, `Product_Name` ASC;
```

- This cte query takes all of the product_ids that have no units in their designated inventory and I added all of the units that are being sold as another column.

- The main query takes the store name, location, and the product name. I ordered it by store_location and product_name

Store_Name	Store_Locati...	Product_Name	Stock_On_Hand	units_sold_in_Mexico
Maven Toys Monterrey 3	Airport	Gamer Headphones	0	15543
Maven Toys Monterrey 3	Airport	Hot Wheels 5-Pack	0	20776
Maven Toys Mexicali 1	Commercial	Action Figure	0	48497
Maven Toys Guanajuato 2	Commercial	Barrel O' Slime	0	54078
Maven Toys Ciudad de Mexico 4	Commercial	Dino Egg	0	28181
Maven Toys Puebla 1	Commercial	Etch A Sketch	0	11205
Maven Toys Guanajuato 2	Commercial	Hot Wheels 5-Pack	0	20776
Maven Toys Puebla 1	Commercial	Hot Wheels 5-Pack	0	20776
Maven Toys Guadalajara 2	Commercial	Hot Wheels 5-Pack	0	20776
Maven Toys Hermosillo 3	Commercial	Jenga	0	12143
Maven Toys Ciudad de Mexico 4	Commercial	Playfoam	0	2812
Maven Toys Hermosillo 3	Commercial	Playfoam	0	2812
Maven Toys Toluca 2	Commercial	Playfoam	0	2812
Maven Toys Guanajuato 2	Commercial	Playfoam	0	2812
Maven Toys Guadalajara 2	Commercial	Plush Pony	0	5328
Maven Toys Saltillo 2	Commercial	Teddy Bear	0	6034
Maven Toys Puebla 1	Commercial	Teddy Bear	0	6034
Maven Toys Mexicali 2	Downtown	Action Figure	0	48497
Maven Toys Xalapa 2	Downtown	Action Figure	0	48497
Maven Toys Puebla 2	Downtown	Animal Figures	0	32250
Maven Toys Culiacan 1	Downtown	Animal Figures	0	32250
Maven Toys Morelia 1	Downtown	Chutes & Ladders	0	3700
Maven Toys Hermosillo 2	Downtown	Chutes & Ladders	0	3700
Maven Toys Mexicali 2	Downtown	Dino Egg	0	28181
Maven Toys Villahermosa 1	Downtown	Dino Egg	0	28181
Maven Toys Monterrey 2	Downtown	Dino Egg	0	28181
Maven Toys Pachuca 1	Downtown	Dino Egg	0	28181
Maven Toys Mexicali 2	Downtown	Etch A Sketch	0	11205
Maven Toys La Paz 1	Downtown	Etch A Sketch	0	11205
Maven Toys Oaxaca 1	Downtown	Etch A Sketch	0	11205
Maven Toys Hermosillo 2	Downtown	Etch A Sketch	0	11205
Maven Toys La Paz 1	Downtown	Foam Disk Launcher	0	6812
Maven Toys Mexicali 2	Downtown	Foam Disk Launcher	0	6812
Maven Toys Tuxtla Gutierrez 1	Downtown	Foam Disk Launcher	0	6812
Maven Toys Aguascalientes 1	Downtown	Foam Disk Launcher	0	6812
Maven Toys Chihuahua 2	Downtown	Foam Disk Launcher	0	6812
Maven Toys Pachuca 1	Downtown	Gamer Headphones	0	15543
Maven Toys Pachuca 1	Downtown	Glass Marbles	0	24507
Maven Toys Puebla 2	Downtown	Hot Wheels 5-Pack	0	20776
Maven Toys Aguascalientes 1	Downtown	Hot Wheels 5-Pack	0	20776
Maven Toys Chilpancingo 1	Downtown	Hot Wheels 5-Pack	0	20776
Maven Toys Xalapa 2	Downtown	Hot Wheels 5-Pack	0	20776
Maven Toys Culiacan 1	Downtown	Hot Wheels 5-Pack	0	20776
Maven Toys La Paz 1	Downtown	Hot Wheels 5-Pack	0	20776
Maven Toys Guanajuato 1	Downtown	Hot Wheels 5-Pack	0	20776

Answer: The location that had the most stores with lost sales was the downtown area. They were missing sales in Action figures, Animal figures, Chutes and Ladders, Dino Eggs, Etch A Sketch, Foam Disk Launchers, Gamer Headphones, Glass Marbles, Hot Wheels 5-packs, Mini Ping Pong Sets, Playfoams, Plush Ponys, Splash Balls, and Toy Robots.

To make the table even more simple, I created another table that collected the amount of stores of each location that had lost sales. I ended up making the original main query as `cte_2`.

```
18
19      SELECT (SELECT COUNT(*) FROM `cte_2` WHERE `Store_Location` = 'Airport') AS `no_Airport`,
20      (SELECT COUNT(`Store_location`)FROM `cte_2` WHERE `Store_Location` = 'Commercial') AS `no_Commercial`,
21      (SELECT COUNT(`Store_Location`) FROM `cte_2` WHERE `Store_Location` = 'Downtown') AS `no_Downtown`,
22      (SELECT COUNT(`Store_Location`) FROM `cte_2` WHERE `Store_Location` = 'Residential') AS `no_Residential`
23
```

no_Airport	no_Commercial	no_Downtown	no_Residential
2	15	46	14

- This way, we can see what location is experiencing the most loss sales.

4. How much money is tied up in inventory at the toy stores? How long will it last?
- Im going to need the store names (for toy stores)
- Product name and cost (i can calculate how much is each product and total them together)
- Stock_on_hand (shows inventory)

I decided to create a double cte, just like the other queries that I did earlier. I would also have to use a JOIN statement as well, since Im using three tables for this analysis (Products, Inventory, stores).

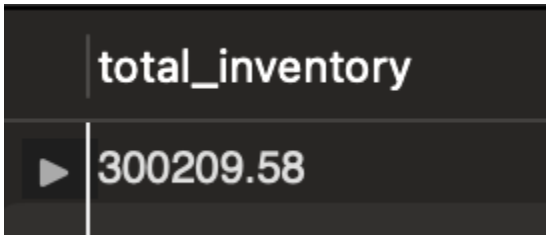
- For part two of the question, Ill just get a list of every inventory of the stores and calculate how long each inventory will last.

```sql
CREATE VIEW toy_sales.`cash_in_inventory` AS
 -- Joins tables `stores, inventory, products.
-- Also, only collects toy stores
WITH `cte` AS (
SELECT st.`Store_Name`,pr.`Product_Name`, pr.`Product_Cost`, i.`Stock_On_Hand`
FROM toy_sales.stores `st`
INNER JOIN
toy_sales.inventory `i`
ON st.`Store_ID` = i.`Store_ID`
INNER JOIN
toy_sales.products `pr`
ON i.`Product_ID` = pr.`Product_ID`
WHERE `Store_Name` LIKE '%toys%'
GROUP BY `Store_Name`, `Product_Name`,`Product_Cost`, `Stock_On_Hand`
ORDER BY `Store_Name` ASC, `Product_Name` ASC),

-- calculates product_cost and the stock_on_hand to find each store's invntory value of a product
 `cte_2` AS
 (SELECT  `Store_Name`,`Product_Name`,`Product_Cost`, `Stock_On_Hand`,
ROUND((`Product_Cost`) * (`Stock_On_Hand`),2) AS `Inventory_value`
FROM `cte`)

-- calculates the total amount of money tied up in inventory at the toy stores in row
SELECT ROUND(SUM(`inventory_value`),2) as `total_inventory`
 FROM `cte_2`
 LIMIT 1
```

- This is a view that joins stores, inventory, and products table
- Collects all the toy stores
- Gets the columns store name, product name, product cost, and the stock_on_hand
- The second cte takes the inventory value (formula = product cost * stock_on_hand) of each product of each store
- The main query takes the sum of all inventory value and round to the nearest hundredth



This is the total inventory in toy stores.

How long will it last?

- I basically took the two ctes, but the main query takes a different formula.

```sql
-- calculates product_cost and the stock_on_hand to find each store's invntory value of a product
, `cte_2` AS (
 SELECT  `Store_Name`,`store_city`,`Product_Name`,`Product_Cost`, `Stock_On_Hand`,
ROUND((`Product_Cost`) * (`Stock_On_Hand`),2) AS `Inventory_value`
FROM `cte`)
-- formula days in inventory = (avg inventory/ cost of goods sold) * period length
-- since theres no data on the average of how much a
SELECT `Store_Name`,`store_city`,ROUND(((`Stock_On_Hand`/`Inventory_value`) * 365),2)AS `days_in_inventory`
FROM `cte_2`
GROUP BY `Store_Name`, `Store_city`, `days_in_inventory`
```

- The formula I did was inventory in days, which was (avg inventory/cost of goods sold) * 365 (365 days)
- I got a whole list of each store of how long their inventory last for

I then just took the average of all of the numbers

```sql
SELECT AVG(`days_in_inventory`) AS `avg_days_in_inventory`
FROM `cte_3`;
```

A: the avg toy store inventory lasts around 52 days