

## 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.

```
1  -- Remove ($) signs from the data, since SQL won't be able to calculate.
2  UPDATE toy_sales.products
3  SET `Product_Cost` = REPLACE(`Product_Cost`, '$', ''),
4      `Product_Price` = REPLACE(`Product_Price`, '$', '');
5
```

### Client Recommendations:

- Which product categories drive the biggest profits? Is this the same across store locations?
- Can you find any seasonal trends or patterns in the sales data?
- Are sales being lost with out-of-stock products at certain locations?
- How much money is tied up in inventory at the toy stores? How long will it last?

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.

```

1 CREATE VIEW toy_sales.`join_one` AS
2
3 -- joins tables products, stores, and sales and put them together as one whole table
4
5 WITH cte AS (
6   SELECT pr.`Product_Name`, pr.`Product_Category`, pr.`Product_Cost`, pr.`Product_Price`, SUM(sa.`Units`) AS total_units,
7     st.`Store_Name`, st.`Store_Location`
8   FROM toy_sales.sales AS sa
9   INNER JOIN
10    toy_sales.products AS pr
11   ON sa.`Product_ID` = pr.`Product_ID`
12   INNER JOIN
13    toy_sales.stores AS st
14   ON sa.`Store_ID` = st.`Store_ID`
15   GROUP BY pr.`Product_Name`, pr.`Product_Category`, pr.`Product_Cost`, pr.`Product_Price`, st.`Store_Name`, st.`Store_Location`
16   ORDER BY `Product_Name` ASC
17 )
18
19 SELECT *, ROUND((total_units * `Product_Price`) - (total_units * `Product_Cost`), 2) AS `profit`
20 FROM cte -- generates table with profit
21 ORDER BY `profit` DESC;

```

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, since this calculates the profit of all the products from every store.

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

Which product categories generate the most product?

```

1 SELECT `Product_Category`, SUM(`profit`) AS `total_profit`
2 FROM toy_sales.join_one
3 GROUP BY `Product_Category`
4 ORDER BY `total_profit` DESC
5

```

	Product_Category	total_profit
►	Toys	1079527
	Electronics	1001437
	Art & Crafts	753354
	Games	673993
	Sports & Outdoors	505718

- The query takes the sum of all profit from each product and then it gets grouped up by it's category, which is know as its `total\_profit`

A: Toys and Electronics are top 2 in generated most generated categories compared to other categories.

Is it the same across all stores????

```

1  -- takes sum of the profit of each product cateogry
2  WITH `cte` AS (
3  | SELECT `Store_Location`, `Product_Category`, SUM(`profit`) AS `total_profit`
4  | FROM toy_sales.join_one
5  | GROUP BY `Store_Location`, `Product_Category`)
6
7  -- takes the category with the highest profit value from each location. It also returns the profit as well.
8  SELECT `Store_Location`, `Product_Category`, `total_profit`
9  FROM `cte`
10 WHERE (`Store_Location`, `total_profit`) IN (SELECT `Store_Location`, MAX(`total_profit`) FROM `cte` GROUP BY `Store_Location`)
11 ORDER BY `Store_Location` ASC;

```

- This is similar query compared to the first one, but this time I get the store name.
- However, this just returns the category with the highest profit value from each store.

	Store_Locati...	Product_Category	total_profit
►	Airport	Electronics	108197
	Commercial	Electronics	287574
	Downtown	Toys	630029
	Residential	Toys	136214

A: The most profitable product for airport and commercial are electronics, while downtown and residential had toys as their most profitable product. Overall, toys still had the highest total profit.

## 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
- Also, I need to find what kinds of seasons does Mexico have
  - Found out that Mexico has three relative seasons (Driest, Low, Shoulder)
- Need products and sales tables (JOIN)

I've learned that Mexico has different seasons related to weather. Instead of just having two seasons, such as rainy and dry, I've found that there is a "shoulder" season, which basically means the transition from Dry to Rainy. So i've done this:

- 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>

```
1  -- a cte that puts it in seasons (driest, shoulder, low)
2  WITH `cte` AS (
3  SELECT *, CASE
4  WHEN MONTH(`date`) IN (12,1,2,3,4) THEN 'DRIEST'
5  WHEN MONTH(`date`) IN (7,8) THEN 'SHOULDER'
6  WHEN MONTH(`date`) IN (5,6,9,10,11) THEN 'LOW'
7  ELSE 'No month available'
8  END AS `seasons`
9  FROM toy_sales.sales
10 ),
11
12 -- gets the prices of the products, products name, and category
13 `cte_1` AS (
14 SELECT pr.`Product_Name`, pr.`Product_Category`, pr.`Product_Cost`, pr.`Product_Price`, SUM(sa.`Units`) AS `total_units`,
15 sa.`seasons`
16 FROM `cte` AS `sa`
17 INNER JOIN
18 toy_sales.products `pr`
19 ON sa.`Product_ID` = pr.`Product_ID`
20 GROUP BY `Product_Name`, `Product_Category`, `Product_Cost`, `Product_Price`, `seasons`),
21
22
23 -- gets the profit for each season and rounds them to the nearest hundreth
24 `cte_2` AS (
25 SELECT `Seasons`, ROUND((`total_units` * `Product_Price`) - (`total_units` * `Product_Cost`),2) AS `profit`
26 FROM `cte_1`
27 GROUP BY `Seasons`, `profit`
28 ORDER BY `profit` DESC)
29
30 -- gets the total profit from each season
31 SELECT `Seasons`, SUM(`profit`) As `total_profit`
32 FROM `cte_2`
33 GROUP BY `Seasons`
34 ORDER BY `total_profit` DESC;
```

- I created a CTE query that takes the months that the sale was collected and put them into their own separated season
- Then I created another CTE that calculated the profit from each product that occurred in their selected season. I also rounded them to the nearest hundredth, just in case some of those calculations didn't translate well into money (Having more than three or equal decimal places)
- Then my main query puts them into a summarization table and totals all of the profit from each season

	Seasons	total_profit
►	DRIEST	1780549
	LOW	1512782
	SHOULDER	720698

A: The Driest season has the highest total profit among the three seasons. It does have a correlation with months such as December and April, which are the months that have Christmas, New years, and Easter, especially since a majority of Mexican citizens are Roman Catholic. Low season is a close second, especially, since Cinco De Mayo is also apart of that season. Shoulder season is the lowest, however there's only two months that occur in that season alone.

There's correlation between the weather in the Driest season, since this is when Mexico is at it's driest, there can be a correlation that there are more people that are more encouraged to go out and shop since the weather is much nicer.

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.

```

1 • CREATE VIEW toy_sales.`zero` AS
2 -- takes store, the product, and the amount of units in the inventory
3 SELECT st.`Store_ID`, st.`Store_Name`, st.`Store_Location`, pr.`Product_ID`, pr.`Product_Name`, i.`Stock_On_Hand`
4 FROM toy_sales.`inventory` AS `i`
5 INNER JOIN
6 toy_sales.`stores` AS `st`
7 ON i.`Store_ID` = st.`Store_ID`
8 INNER JOIN
9 toy_sales.`products` AS `pr`
10 ON i.`Product_ID` = pr.`Product_ID`
11 INNER JOIN
12 toy_sales.`sales` AS `sa`
13 ON i.`Store_ID` = sa.`Store_ID`
14 WHERE `Stock_On_Hand` = 0
15 GROUP BY `Store_ID`, `Store_Name`, `Store_Location`, `Product_ID`, `Product_Name`, `Stock_On_Hand`
16 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'.

```

1 • WITH `cte` AS (
2 SELECT pr.`Product_ID`, pr.`Product_Name`, SUM(sa.`Units`) AS `units_sold_in_Mexico`
3 FROM toy_sales.sales `sa`
4 INNER JOIN
5 toy_sales.products `pr`
6 ON sa.`Product_ID` = pr.`Product_ID`
7 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)
8 GROUP BY `Product_ID`, `Product_Name`)
9 ,
10 -- creates a table combining with the cte, which puts the store name together, with the stock = 0.
11 `cte_2` AS (
12 SELECT z.`Store_Name`, z.`Store_Location`, z.`Product_Name`, z.`Stock_On_Hand`, `cte`.`units_sold_in_Mexico`
13 FROM toy_sales.zero `z`
14 INNER JOIN
15 `cte`
16 ON z.`Product_ID` = `cte`.`Product_ID`
17 GROUP BY `Store_Name`, `Store_Location`, `Product_Name`, `Stock_On_Hand`, `units_sold_in_mexico`
18 ORDER BY `Store_location` ASC, `Product_Name` ASC)
19
20 -- Occurences of lost sales from each region
21 SELECT (SELECT COUNT(*) FROM `cte_2` WHERE `Store_Location` = 'Airport') AS `no_Airport`,
22 (SELECT COUNT(`Store_Location`) FROM `cte_2` WHERE `Store_Location` = 'Commercial') AS `no_Commercial`,
23 (SELECT COUNT(`Store_Location`) FROM `cte_2` WHERE `Store_Location` = 'Downtown') AS `no_Downtown`,
24 (SELECT COUNT(`Store_Location`) FROM `cte_2` WHERE `Store_Location` = 'Residential') AS `no_Residential`
25

```

- 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' Silime	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 Ponies, 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

```

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

no_Airport	no_Commercial	no_Downtown	no_Residential
2	15	46	14

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.

```

1 CREATE VIEW toy_sales.`cash_in_inventory` AS
2   -- Joins tables `stores`, `inventory`, `products`.
3   -- Also, only collects toy stores
4   WITH `cte` AS (
5     SELECT st.`Store_Name`, pr.`Product_Name`, pr.`Product_Cost`, i.`Stock_On_Hand`
6     FROM toy_sales.stores `st`
7     INNER JOIN
8     toy_sales.inventory `i`
9     ON st.`Store_ID` = i.`Store_ID`
10    INNER JOIN
11    toy_sales.products `pr`
12    ON i.`Product_ID` = pr.`Product_ID`
13    WHERE `Store_Name` LIKE '%toys%'
14    GROUP BY `Store_Name`, `Product_Name`, `Product_Cost`, `Stock_On_Hand`
15    ORDER BY `Store_Name` ASC, `Product_Name` ASC),
16
17   -- calculates product_cost and the stock_on_hand to find each store's inventory value of a product
18   `cte_2` AS
19   (SELECT `Store_Name`, `Product_Name`, `Product_Cost`, `Stock_On_Hand`,
20    ROUND((`Product_Cost`) * (`Stock_On_Hand`),2) AS `Inventory_value`
21    FROM `cte`)
22
23   -- calculates the total amount of money tied up in inventory at the toy stores in row
24   SELECT ROUND(SUM(`Inventory_value`),2) as `total_inventory`
25   FROM `cte_2`
26   LIMIT 1
27

```

- 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

	total_inventory
▶	300209.58

How long will it last?

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

```

14  -- calculates product_cost and the stock_on_hand to find each store's inventory value of a product
15  , `cte_2` AS (
16    SELECT `Store_Name`,`store_city`,`Product_Name`,`Product_Cost`,`Stock_On_Hand`,
17    ROUND((`Product_Cost`) * (`Stock_On_Hand`),2) AS `Inventory_value`
18  FROM `cte`)
19  -- formula days in inventory = (avg inventory/ cost of goods sold) * period length
20  -- since theres no data on the average of how much a
21  SELECT `Store_Name`,`store_city`,ROUND(((`Stock_On_Hand`/`Inventory_value`) * 365),2)AS `days_in_inventory`
22  FROM `cte_2`
23  GROUP BY `Store_Name`,`Store_city`,`days_in_inventory`
24

```

- 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

```

5  SELECT AVG(`days_in_inventory`) AS `avg_days_in_inventory`
6  FROM `cte_3`;

```

A: the avg toy store inventory lasts around 52 days