OBA444: Project Report

Video games, Accessories and Consoles

Trenten Meyer, Elijah Raffo, Johnny Whitaker

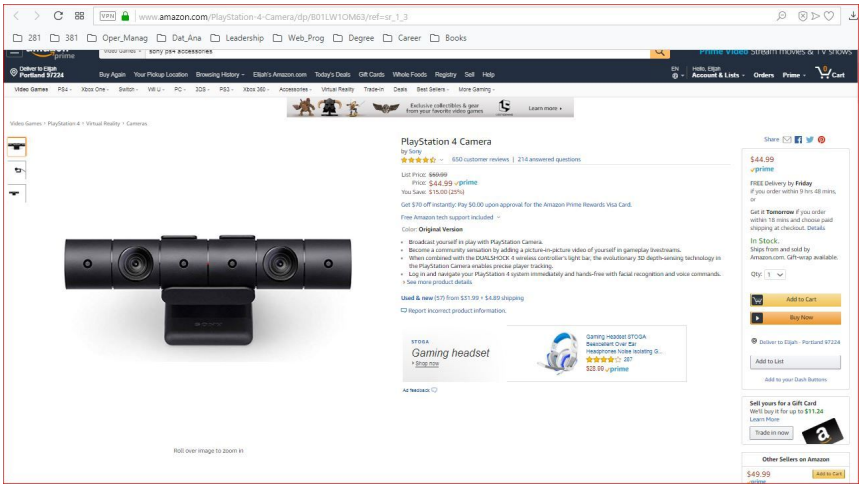
OBA444 T/Th 10:00AM

Erin Cil

March 17 2019

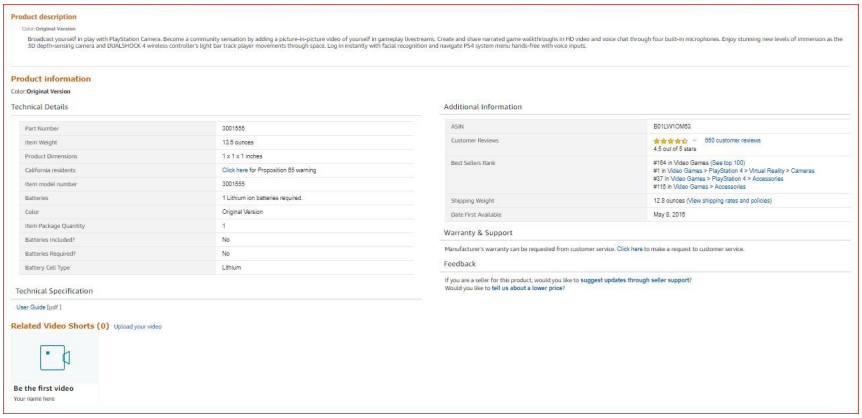
For our project, we decided on studying video games, Within the video games segment, we would specifically look at the games, accessories, and consoles. We chose this category because it is a growing industry with profit potential. This report explains the creation of our database, entity relationship diagram, relational schema, and the four queries ran in our database.

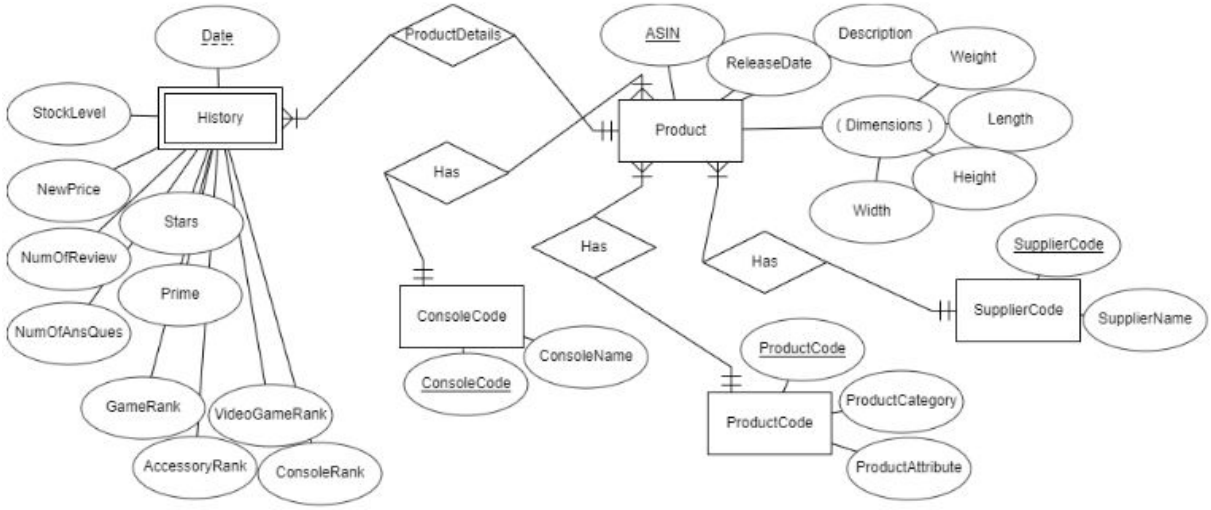
Entity Relationship Diagram:



Our database for capturing information on Video Game derives from amazon screen shots. Captured information is divided into three categories: Consoles, Accessories, and Games. We had ~15 products per category.

We had five data collection points over ten days (02.26 to 03.06). Our data consisted of capturing two screenshots for every product. To the right you are two screenshots. The top screenshot shows the general information, and the bottom screenshot is of the product description. Our captured data enabled us to create a database comprised of five tables: History, Product, Console Code, Product Code, and Supplier Code. All fields in our database are mandatory. At the bottom the next page you can see the ER diagram.





Our history table is an analytical table and a weak entity. The weak primary key for this table is defined by the date we capture data for the entry. We captured data once every other day on each product. For each history table entry we record for the day data was collected, whether something is in or out of stock, the new price of the item, number of reviews, number of answered questions, star rating, whether the item qualifies for prime shipping, general ranking, and category ranking. A majority of this data is captured in the general screenshot.

General ranking and category ranking comprise of 4 fields: Video Game ranking, Accessory Ranking, Game Ranking, and Console Ranking. This information is detailed in the product description screenshot. Video game ranking was qualified as the ranking for the "Video Games" category in amazon. Depending on the product different types of rankings could be detailed. If there was no ranking that fit our description the ranking would be 0. For example, a game could have a video game ranking, game ranking, no console ranking, and no accessory ranking. We would record two ranks, and two zeros. For example, an accessory could have no video game rank, game rank, console rank, or accessory rank within the video game category; however, it would have an electronics rank. We would record that all rankings are zero because the ranking specified is not the category of ranking we are collecting data on.

Our product table is an operational table whose information changes rarely. Each product has a unique ASIN number. Each product has a release date, description, and ship dimensions. Ship dimensions are determined by width, length, height, and weight. A majority of this data is captured in the product description screenshot.

The console code table has the unique identifier of console code. A console code predicts the console name. A majority of this data is captured in the general screenshot.

The product code table has the unique identifier of product code. Each product code predicts the product category and product attribute. A majority of this data is captured in the product description screenshot.

The supplier code table has the unique identifier of supplier code. A supplier code predicts the supplier name. A majority of this data is captured in the general screenshot.

Every other day we enter data into the history table for each product. A product ASIN can be in the history table more than once, but each entry into the history table must have one and only one ASIN from the product table.

Each product in the product table one and only one console code, supplier code, and product code. A product can share a console code, supplier code, and product code. No product can have a console code, supplier code, or product code that is not in the coordinating tables.

Relational Schema Diagram:

The relational schema for our database (pictured below) also shows the relationship between our tables mentioned above and exhibits how future data points/variables could be added to make the database more

robust. We chose are data types within our database to increase the efficacy of analytical information gleaned from querying the database. The schema was created using MySQL and the specific notation used in illustrating the relationships is that a solid line between tables indicates an identifying relationship, whereas a dashed line indicates a non-identifying relationship.

An identifying relationship means that the 'child' table cannot be uniquely identified without the 'parent' table; in our database this is only seen between the Product Table (parent) and the History Table (child). This relationship is defining because the History Table uses a composite primary key including the ASIN number which connects to the Product Table and an end user cannot determine the full characteristics of a product from the History Table alone; full product characteristics necessitate connecting the ASIN numbers. By contrast, the dashed lines between Product Table and Console Code for example, indicate that a Console Name can be uniquely identified by the Console Code without having to depend on a connection to the Product Table.

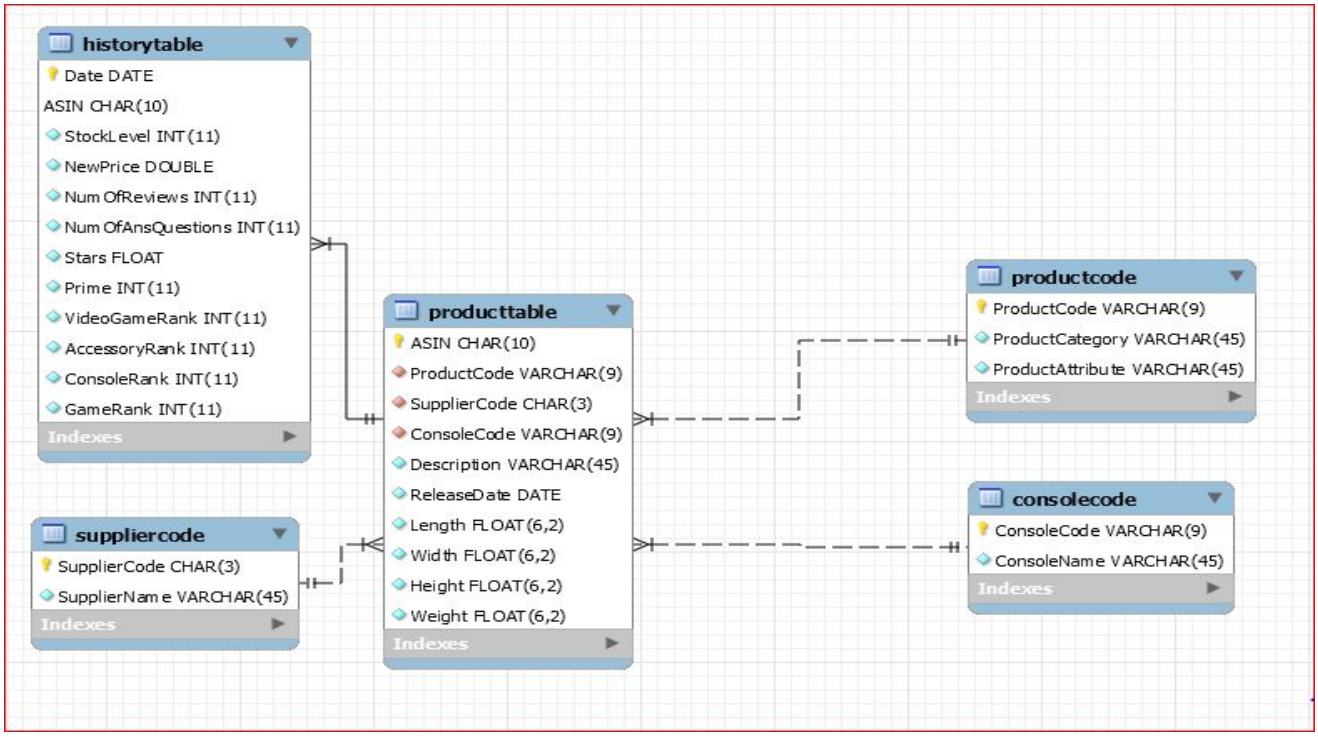
The opportunities for extra data points exist primarily in the Product, Supplier, and Console Code tables with non-identifying relationships. Examples of pertinent data to add could be Supplier characteristics such as global rank by sales, total sales, nation of origin, years in business, etc. Similar opportunities exist for Console characteristics and Product Code characteristics which could increase the efficacy of querying the database for actionable analytical data.

The relational schema below is normalized to 2nd Normal Form. Particular to the 2NF, partial key dependencies are of interest. No partial key dependencies can exist on Supplier Code, Product Code, Console Code, or Product Table as they only have a single primary key. The only opportunity for partial key dependencies exists in the History Table which features a composite primary key of Date of observation and ASIN of the product observed. However, there are 50 entries (one for each product) for each date and 5 entries for each ASIN (one for each day of data collection) and all of the non-primary variables are specific (though not necessarily unique) to the specific combinations of ASIN and Date meaning both primary keys are necessary to uniquely identify all variables, so no partial dependencies exist in our database.

Finally, the relational schema displays the data types for each variable. INT (11) is used for most fields on the History table which are numeric without the inclusion of decimal points; Num Of Reviews, Num Of Ans Questions, Video Game Rank, Accessory Rank, Console Rank, and Game Rank. However, two of the INT (11) variables on the History table are actually binary variables; whether or not a product is Prime (1 - yes, 0 - no), and StockLevel (1 - in stock, 0 - out of stock). NewPrice and Stars are both decimal values; DOUBLE and FLOAT respectively due to the facts that price includes multiple decimal values and Stars ratings only include a single decimal point. Finally on the history table, ASIN (Amazon Standard Identification Number) is a VARCHAR (10) as it is ubiquitously a 10 character alphanumeric value and the Date is intuitively inputted as DATE.

The Product Code, Console Code, and Supplier Code tables feature VARCHAR (3 or 9) to accommodate our created code values and VARCHAR (45) for the inputting of text attributes of up to 45 characters.

The Product Table features FLOAT values for all four shipping dimension variables to accommodate minimal decimal values, VARCHAR (45) for text in product Description and intuitively uses DATE for Release Date. The other four variables, ASIN, Console Code, Product Code, and Supplier Code feature the same data types as on other tables. Where possible, minimal value space was permitted to mitigate memory usage. The selective utilization of FLOAT vs DOUBLE, the latter of which allots double the memory of the former, and VARCHAR (3) vs (10) or (45) can have a significant impact on the efficacy of the database, particularly should the amount of data entries scale exponentially.



Queries:

```
1 •  SELECT pt.ConsoleCode, count(ht.ASIN) Entries, round(avg(ht.NewPrice),0) Price,
2        round(avg(ht.NumOfReviews),2) Reviews,round(avg(ht.VideoGameRank),0) Ranking
3    FROM historytable ht, producttable pt
4    WHERE ht.ASIN = pt.ASIN and pt.productcode like "GA%"
5    GROUP BY pt.ConsoleCode;
```

ConsoleCode	Entries	Price	Reviews	Ranking
PS4	35	45	284.26	335
SWITCH	15	42	647.93	185
XONE	25	43	63.12	783

For our first query, we were concerned at looking at video games. We wanted to display number of entries, average new price, average number of reviews and average video game ranking for games and group our data by console code. We need the history and product table and joined those tables by linking their ASIN numbers. Also pt.productcode

like “GA%” means we are only looking at games, not accessories or consoles. As you can see from the results, Switch had the least number of entries but lead the way in every other category.

```
SELECT sc.SupplierName, round(avg(ht.Stars),2) AvgStar, round(avg(ht.NumOfReviews),0) AvgReviews,
    round(avg(ht.NumOfAnsQuestions),0) AvgNumAnsQuest, round(avg(ht.VideoGameRank),0) AvgRank
FROM historytable ht, producttable pt, suppliercode sc
WHERE ht.ASIN = pt.ASIN and sc.SupplierCode = pt.SupplierCode and ht.VideoGameRank < 1000 and ht.VideoGameRank <>0
GROUP BY sc.SupplierName
ORDER BY SupplierName ASC;
```

SupplierName	AvgStar	AvgReviews	AvgNumAnsQuest	AvgRank
Bandai Namco	4.16	70	33	427
Microsoft	3.78	965	192	252
Nintendo	4.53	1275	196	186
Seagate	4.30	1681	530	92
Sony	3.72	2085	365	148
Ubisoft	2.27	162	40	180
WB Games	3.74	101	58	449

For our second query, we are only concerned about Suppliers. We wanted to display the supplier name, average number of stars, average number of reviews, average number of questions answered, and average video game rank where video game rank must be less than 1000 and not equal to 0. We limited the Rank because extreme values skewed the results. We also need the history, product, and supplier code tables and joined them by linking ASIN number for history and product and supplier code for product table and supplier code table.We also want to group by suppliers and order by the supplier name in ascending order.

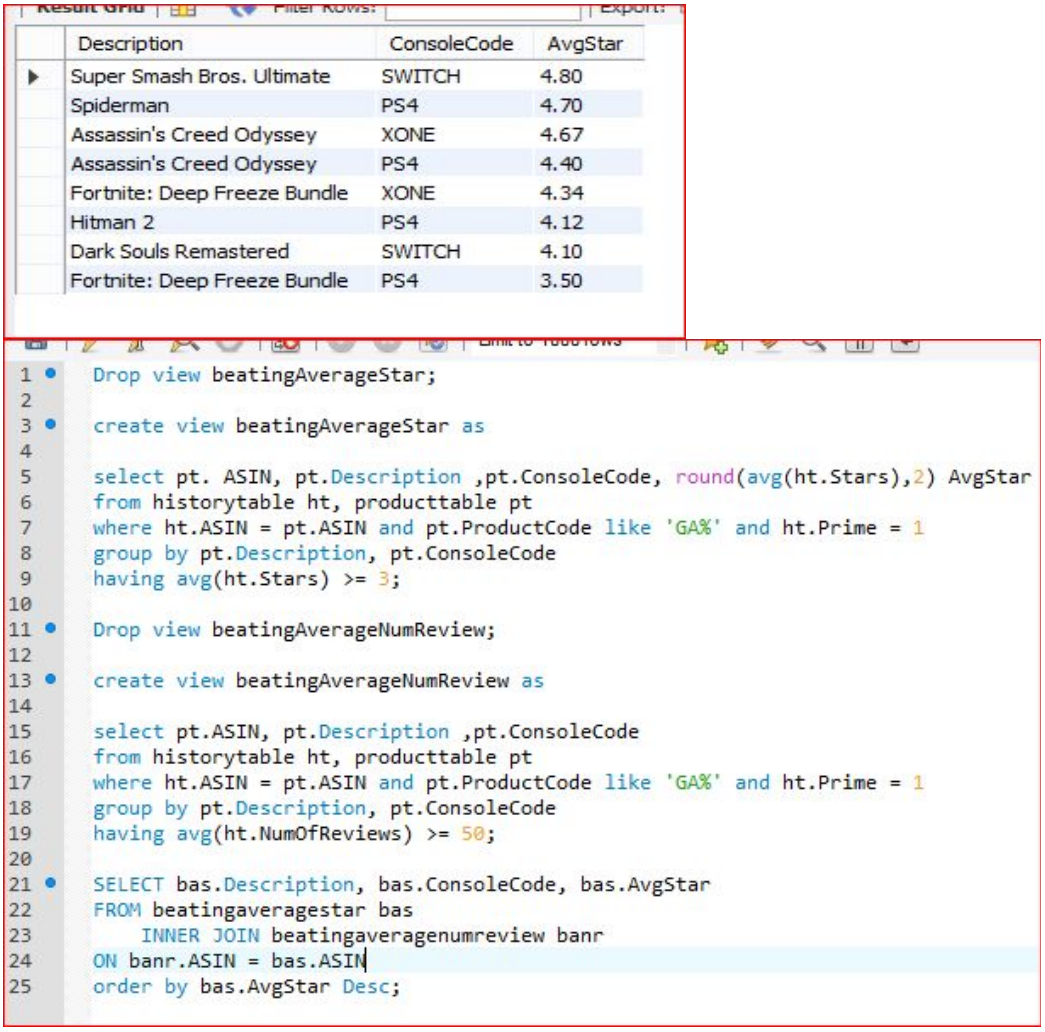
```
SELECT pc.ProductAttribute, pt.ConsoleCode, round(avg(ht.NewPrice),2) AvgPrice
FROM historytable ht, producttable pt, productcode pc
WHERE ht.ASIN = pt.ASIN and pc.ProductCode = pt.ProductCode and
(pc.ProductCode = 'AC_CON' or pc.ProductCode = 'AC_CSL+')
GROUP BY pc.ProductAttribute, pt.ConsoleCode
order by pc.ProductAttribute Desc, AvgPrice Asc
```

Controller	PS4	46.59
Controller	SWITCH	48.52
Console Add-On	XONE	17.97
Console Add-On	SWITCH	45.88
Console Add-On	PS4	54.42

For our third query, we were interested in looking at accessories, specifically controllers and console add-ons. We wanted to display the product attribute, the console code, and average new price (display as AvgPrice). We need the history, product, and product code tables and joined them by linking ASIN number for history and product table and product code for product and product code tables. We restricted product code to ‘AC_CON’ or ‘AC_CSL+’ to only show controllers or console add ons. We then group by product attribute and console code. Order by descending product attribute and ascending average new price for easy readability.

For our final query, we were interested in doing a view and intersection. We wanted to create two separate views and then used an intersection to combine them. The first view displays for games the average stars, console code, ASIN, and description of video games that are prime eligible and have an

average star rating of greater than 3. We need the history and product table for this view and joined them by linking ASIN numbers. Then we grouped that view by description and console code. The second view for games displays ASIN, Description, and console code for games that are prime eligible and have at least 50 or more reviews and are grouped by description and console code. For this view we need again the history and product tables and joined them by linking ASIN numbers. We then used an inner join statement to connect the two views. We displayed description, console code, and average stars and ordered by descending average stars.



Description	ConsoleCode	AvgStar
Super Smash Bros. Ultimate	SWITCH	4.80
Spiderman	PS4	4.70
Assassin's Creed Odyssey	XONE	4.67
Assassin's Creed Odyssey	PS4	4.40
Fortnite: Deep Freeze Bundle	XONE	4.34
Hitman 2	PS4	4.12
Dark Souls Remastered	SWITCH	4.10
Fortnite: Deep Freeze Bundle	PS4	3.50

```
1  Drop view beatingAverageStar;
2
3  create view beatingAverageStar as
4
5  select pt. ASIN, pt.Description ,pt.ConsoleCode, round(avg(ht.Stars),2) AvgStar
6  from historytable ht, producttable pt
7  where ht.ASIN = pt.ASIN and pt.ProductCode like 'GA%' and ht.Prime = 1
8  group by pt.Description, pt.ConsoleCode
9  having avg(ht.Stars) >= 3;
10
11 Drop view beatingAverageNumReview;
12
13 create view beatingAverageNumReview as
14
15 select pt.ASIN, pt.Description ,pt.ConsoleCode
16 from historytable ht, producttable pt
17 where ht.ASIN = pt.ASIN and pt.ProductCode like 'GA%' and ht.Prime = 1
18 group by pt.Description, pt.ConsoleCode
19 having avg(ht.NumOfReviews) >= 50;
20
21 SELECT bas.Description, bas.ConsoleCode, bas.AvgStar
22 FROM beatingaveragestar bas
23     INNER JOIN beatingaveragenumreview banr
24 ON banr.ASIN = bas.ASIN
25 order by bas.AvgStar Desc;
```

Through this process, we all learned a lot about data collection, creating and populating a database, and then how to derive meaningful information from that database. I think we underestimated the time it can take to collect data and overestimated the simplicity of creating queries once we had our database populated. Our group felt this was a rewarding project as it can be scaled in many different ways and be useful to both consumers and businesses, all depending on what queries are ran and what information is trying to be derived.