Partnership Analyst Case Study

Created: December 2024
Created By: Robert (Dale) Earnhardt
Assigned: Christian Granados
Submission: Friday, January 17th, 2025

Part 1: Data Modeling

Data Ingestion

The initial EDA I conducted was done using Python, Pandas, and Numpy within an IDE (Visual Studio Code). I choose this versus EDA with SQLite because Pandas offered a more powerful way to initially explore and model the data. All cleaning, standardization, and data model creation was done in SQLite as well. I chose this flavor of SQL because I am only working with local CSVs; SQLite excels for high-performance, self-contained scenarios.

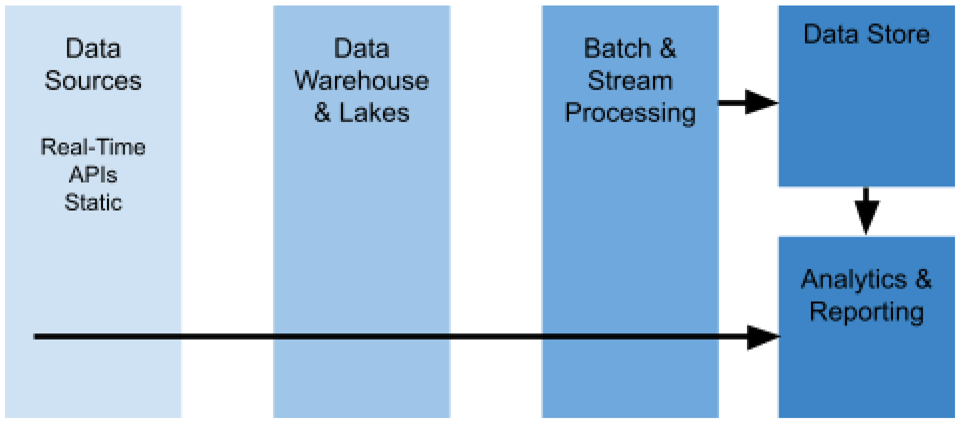
To load the CSVs into a Jupyter Notebook I created a new directory for the project with all relevant files. No chunking or batch processing was needed due to the relatively small amount of data this case entails. Within the notebook ,since the CSVs were in my working directory, I loaded them into Pandas using the following:

```
import pandas as pd
bc_df = pd.read_csv('broadcast_data_set.csv')
sc_df = pd.read_csv('social_data_set.csv')
```

To load the data into SQLite3 database I used the SQLite3 Python package. I created a local database, created a custom table, and pre-processed the data below before insertion. I decided on using a SQLite3 database within Python because (1) I am already familiar with the editor and (2) all commands can be seen within the python file.

1) Switch from UTF-8 BOM to UTF-8 encoding
1) Pre-processed by replacing empty string values with NULL to avoid data type mismatches. Then, specifying these NULL values when importing.
2) Convert dates to UNIX time because SQLite3 does not natively support dates, timestamps, or datetimes.
3) Dropping rows that have NULL entries for Non-Null fields. This is determined through my EDA, data definitions, and usefulness in analysis.
    a) This was done by adding a NOT NULL constraint with a conflict resolution type of IGNORE.

The above method significantly differs from how a company would perform ingestion.

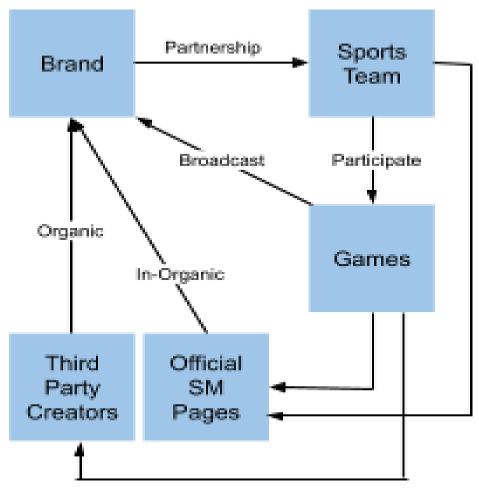


If this case was to exclusively be accomplished in the cloud via Google Cloud Platform (GCP). The files would be placed in a Cloud Data Bucket. They then would be queried via Cloud SQL into more permanent tables. From here there are multiple methods for analysis:

1) Utilizing Vertex AI Workbenches would allow flexible exploration of the data via a Jupyter Notebook
2) Cloud run could be utilized to run predefined analysis scripts or host data web-apps
3) One could query inside the Cloud SQL database to get relevant data

Data Description

A large portion of my time spent on this case was dedicated to understanding the data. Besides the given materials – such as the datasets, case document, and information provided in previous interviews – I also leveraged asking questions to Michelle and Dale, online research of other sports analytics practices, data-centric technical blogs, published Two Circles case studies, and general experimentation. However, at the end of the day there is always ambiguity and the definitions below are a result of my best assessment.



I am defining a 'leading property within one of the North American sports leagues' as a professional sports team. They have sponsorship deals to promote brands; brands are promoted by having their logo or recognizable signage on visible parts of the games. This can be virtual – such as transitions or on-screen pop-ups – or physical – such as logos on basketball backboards. The social_dataset concerns this branding showing up in social media posts on the internet. This can take the form of first-party posts from official channels or third-party posts from non-affiliated creators. The broadcast_dataset concerns this branding showing up during the streaming of these games.

Social Media Dataset - Data Definitions

Post_Id: A unique identifier for each social media post.
Exposure_Id: Since a social media post can contain multiple brands, this unique identifier differentiates between brands.
Content_Type: The form of the social media post. (Photo, Video, or Text.)
Channel: The social media platform. (Facebook, Twitter, YouTube)
Partner: Brand of the featured sponsored content
Partner_Exposure_Date: The day the post went live
Snapshot_Date: For the given post, the date at which the social media analytics data was recorded.
Feed_Name: Binary variable that details if a post is made by a first party (Owned) or a third party (Earned).
Asset: Non-exhaustive categorical variable about what the branded content appears on. (i.e. Scoreboard, Seatbacks)

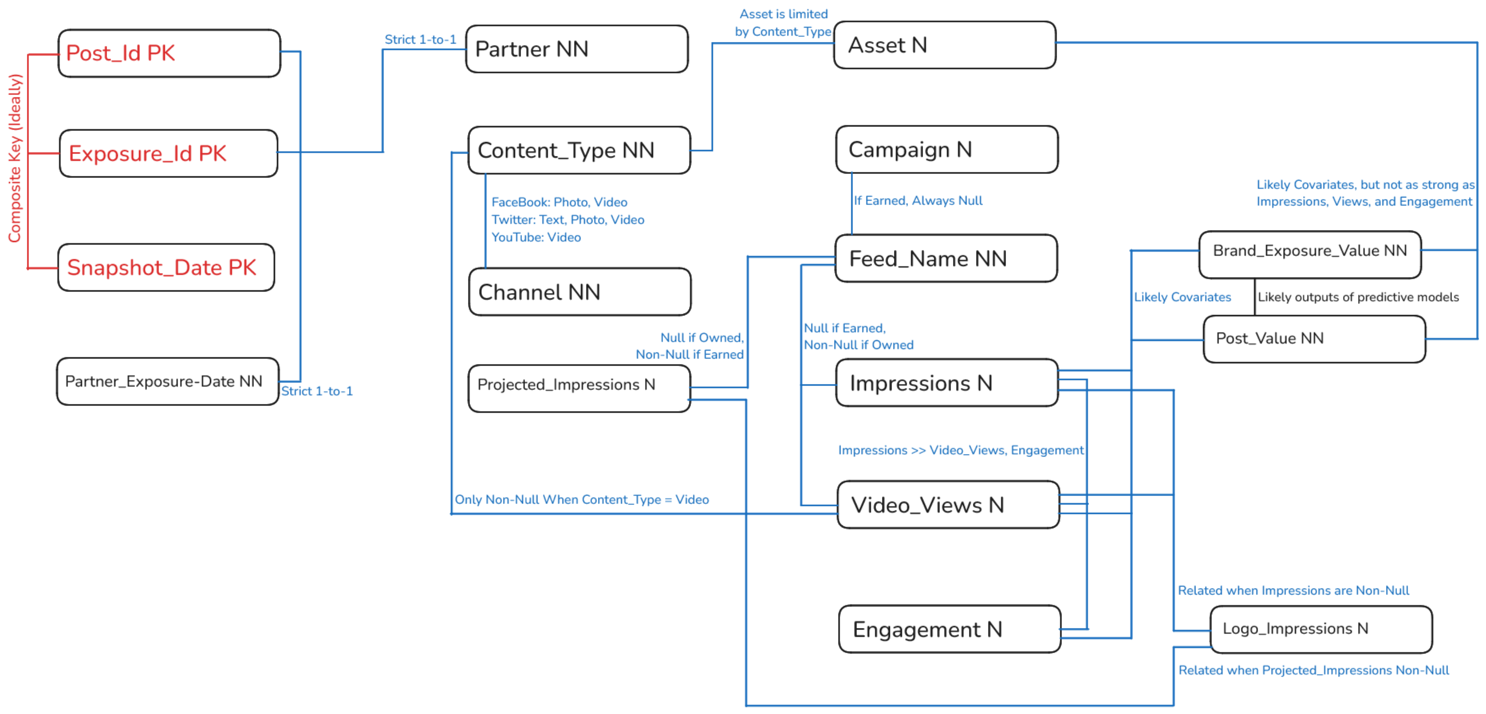
Post_Value, Brand_Exposure_Value: Likely outputs of a predictive model that is tasked with determining an exposure's 'value.' Social media analytics, such as impressions and engagement, are almost certainly factors but do not tell the whole story.
Campaign: Stores if a post was part of a marketing campaign and if so, what campaign one.
Impressions: Defined according to each social media platform but is usually the number of times a post was visible to a user.
Video_Views: Number of times a user viewed a piece of video content. Calculations are dependent on social media, but usually requires a minimum watch watch time to be counted as a 'view'.
Engagement: Specific to each social media platform, but is usually the sum of all interactions users have had with a post. (i.e. Like, Comment, Share)
Projected Impressions: In the event Two Circles does not have access to analytics (i.e. third-party content) or analytics are no longer available, a proxy for impressions is used.

Logo Impressions were not featured in the above table and this is for a reason. Through my EDA I found that it was almost always a rescaling of Projected Impressions divided by 5.7 or equal to Impressions. For different segments of posts, the below table shows what Logo_Impressions is equal to:

	Non-Null Impressions	Non-Null Projected Imp
Twitter, Video	Views	Projected_Impressions/5.7
Facebook, Video	N/A	Projected_Impressions/5.7
YouTube, Video	Impressions, Views	N/A
Facebook, Photo	N/A	Projected_Impressions/5.7
Twitter, Photo	Impressions	Projected_Impressions/5.7
Twitter, Text	N/A	N/A

Social Media Dataset - Relationships

This visual represents the relationships between variables I discovered during EDA.This is a formatted version of the picture in the appendix.



Broadcast Dataset

Season: The season the NHL game took place in
Date: Date of the game
Game: Opponentsof the given game
Brand: Partnership brand; analogous to partner in social_dataset
Asset: Non-exhaustive categorical variable about what the branded content appears on. (i.e. Scoreboard, Seatbacks) Analogous to social_dataset
Exposure: Number of times the sponsor was shown in the broadcast through this branded asset
Duration: Total time this branded asset was seen during the game (seconds)
QIMV: Quality Impressions per Minute Viewed. An externally calculated metric that estimates the quality of the brand exposure
Exposure_Impressions: An estimate or proxy of total impressions based on exposure and possible other features
Sponsorship_Impressions: Impressions directly attributed to the branded sponsorship placement

QI_Sponsorship_Impressions: Quality impressions, determined by external criteria, attributed to branded sponsorship placement

Data Cleaning & Standardization

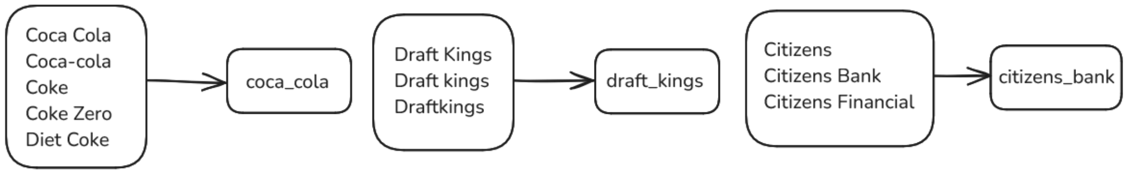
No Primary Key

Originally I thought the Exposure_Id recorded a unique instance of a piece of branded content showing up in a post. After further inspection, this is wrong. The Exposure_Id only related to the brand itself, not the Asset. This is an issue because to create a natural primary key, Asset would need to be included. Asset contains Null values; by definition, a Primary Key cannot contain Null values.

To alleviate this issue one could either transform Asset to restrict it to Non-Null values or Exposure_Id could be modified for all duplicate rows. Modifying Exposure_Id is harder but more robust. For example, non-duplicate branding could occur on the same asset for the same company. Additionally, since Asset is a non-exhaustive categorical labeling, two assets could be different while both are encoded as NULL in the table.

Brand Names

In both tables, the brand names had multiple variations, misspelling, inconsistent spacing, and many more issues. I standardized them by grouping together, renaming entries, and converting all naming to lower snake case.

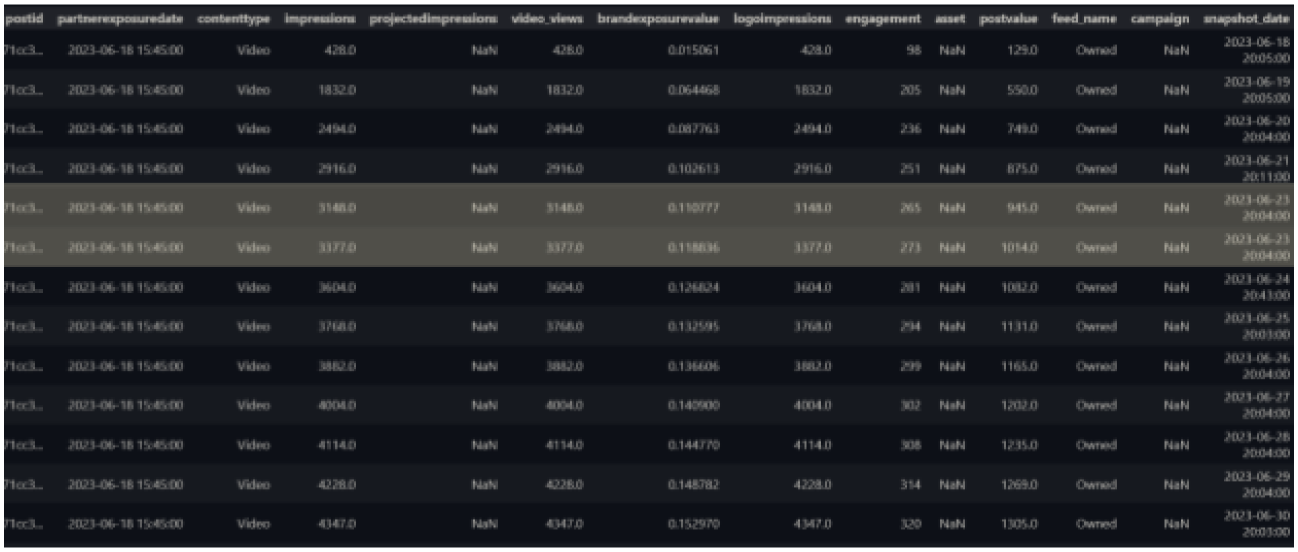


Asset, Campaign, Game Names

Similar to brand/partner naming, asset and campaign naming was inconsistent within each table and when cross-referencing between tables. Game entries only existed in the broadcast_dataset, but still had within-table inconsistencies. As above I grouped together entries that described the same asset and converted them to lower snake cases.

Dropping Duplicates

Within the social_dataset, for matching post_id, exposure_id, asset, partner and snapshot_date there were entries that had only slight variations of impression, video_views, and engagement values. In the order of a few impressions when dealing with multiple hundreds of thousands. My theory is that these occurred due to multiple, concurrent requests sent out at the same time for the same data. Another idea is the original request could have been taking too long, so an automated retry occurred. This process was also done on Broadcast_data using matching game, brand, asset, and exposure columns.



postid	partnerexposuredate	contenttype	impressions	projectedimpressions	video_views	brandexposurevalue	logoimpressions	engagement	asset	postvalue	feed_name	campaign	snapshot_date
71cc3...	2023-06-18 15:45:00	Video	428.0	NaN	428.0	0.015061	428.0	98	NaN	129.0	Owned	NaN	2023-06-18 20:05:00
71cc3...	2023-06-18 15:45:00	Video	1832.0	NaN	1832.0	0.064468	1832.0	205	NaN	550.0	Owned	NaN	2023-06-19 20:05:00
71cc3...	2023-06-18 15:45:00	Video	2494.0	NaN	2494.0	0.087763	2494.0	236	NaN	749.0	Owned	NaN	2023-06-20 20:04:00
71cc3...	2023-06-18 15:45:00	Video	2916.0	NaN	2916.0	0.102613	2916.0	251	NaN	875.0	Owned	NaN	2023-06-21 20:11:00
71cc3...	2023-06-18 15:45:00	Video	3148.0	NaN	3148.0	0.110777	3148.0	265	NaN	945.0	Owned	NaN	2023-06-23 20:04:00
71cc3...	2023-06-18 15:45:00	Video	3377.0	NaN	3377.0	0.118836	3377.0	273	NaN	1014.0	Owned	NaN	2023-06-23 20:04:00
71cc3...	2023-06-18 15:45:00	Video	3604.0	NaN	3604.0	0.126824	3604.0	281	NaN	1082.0	Owned	NaN	2023-06-24 20:43:00
71cc3...	2023-06-18 15:45:00	Video	3768.0	NaN	3768.0	0.132595	3768.0	294	NaN	1131.0	Owned	NaN	2023-06-25 20:03:00
71cc3...	2023-06-18 15:45:00	Video	3882.0	NaN	3882.0	0.136606	3882.0	299	NaN	1165.0	Owned	NaN	2023-06-26 20:04:00
71cc3...	2023-06-18 15:45:00	Video	4004.0	NaN	4004.0	0.140900	4004.0	302	NaN	1202.0	Owned	NaN	2023-06-27 20:04:00
71cc3...	2023-06-18 15:45:00	Video	4114.0	NaN	4114.0	0.144770	4114.0	308	NaN	1235.0	Owned	NaN	2023-06-28 20:04:00
71cc3...	2023-06-18 15:45:00	Video	4228.0	NaN	4228.0	0.148782	4228.0	314	NaN	1269.0	Owned	NaN	2023-06-29 20:04:00
71cc3...	2023-06-18 15:45:00	Video	4347.0	NaN	4347.0	0.152970	4347.0	320	NaN	1305.0	Owned	NaN	2023-06-30 20:03:00

Logo Impressions
For the social_dataset, an anomaly I found was that for photos posted on Twitter (with non-null impressions) a large portion of the Logo_Impressions were equal to impressions divided by 5.7, and not impressions themselves. This was corrected.

Views greater than Impressions
In some rows in social_dataset, the Impressions were less than Video_Views. Based on definitions from social media platforms this should not be possible. If a person views a piece of content, it is guaranteed it has already appeared on their screen. Impressions were raised to equal Video_Views in these scenarios.

Modeled Output for Reporting

Data Combination

Prior to broadcast_dataset being joined to social_dataset I did some pre-processing. I decided to join broadcast_dataset in because it was the smaller table. I did not join on a common key but rather concatenated the tables and combined repeated columns.

Exposure_Id

Since each row will be a unique exposure, this acts as the primary key and is created by a randomly generated 56-character UUID.

Post_Id
Similarly to the social_dataset, the same Post_Id will exist for each unique, shared piece of content. In this instance, the games themselves. This means the same game will need to have the same Post_id. I applied the following function to the game column to create a 56-character hash with the game as a seed value.

```
import hashlib
broadcast_df['postid'] = broadcast_df['game'].apply(
     lambda s : hashlib.sha256(str.encode(s)).hexdigest()[:56])
```

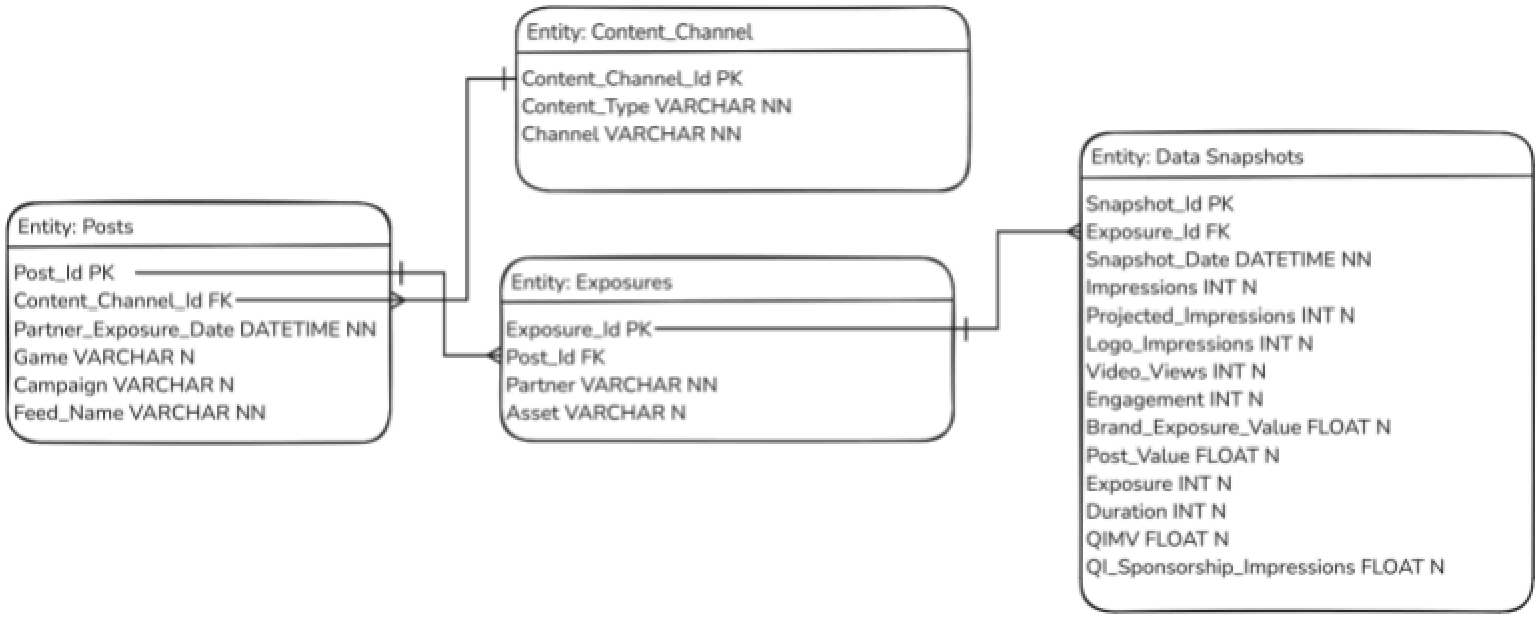
Partner_Exposure_Date & Snapshot_Date
I created these columns from the original date column. Partner_Exposure_Date was set equal to date because if we are treating the whole broadcast as a "post," the brand is first exposed when the programming comes on. Snapshot_Date was set to 2.5 hours after date, because this is when a game usually ends and data is collected.

Channel, Content_Type, Feed_Name
These were set to the corresponding static values of broadcast, video, and owned. I created a unique channel for the broadcast as it does not fit into any of the existing ones, and is largely streamed. Feed_name is defined as owned as the competing team likely has a relationship with the broadcast company for viewership data.

Renaming
I renamed exposure_impressions to impressions and sponsorship_impressions to video_views. From my understanding exposure_impressions is an estimate for total impressions. Sponsorship_impressions is defined above as the impressions directly attributable to the sponsorship placement and is smaller than impressions. Similarly, video_views is a more restrictive version of impressions. Having impressions and video_views as non-null together for a video content_type is consistent with how social_dataset is defined.

Data Model

The normal forms of databases are useful for CRUD operations because they give guarantees on data integrity. If a database is essential for tracking employee information, then maintaining the highest level of normalization and data integrity is paramount. However, the purpose of this table is reporting and analysis. The act of joining disparate datasets, such as the social_data_set and broadcast_data_set, directly violates 5NF. Additionally, the data present looks to be logging information – where data is mainly added only – and is not operationally essential. I have balanced both normalization and ease of analysis principles to create the following table. To communicate it I created the following ERD diagram.



I decided to break up the data into three main entities: Posts, Exposures, and Data Snapshots. At a high level my main goal was to create an easier analysis process by reducing data redundancy. The Posts entity is defined as a unique form of shareable and viewable content. This definition also allows me to treat each game featured in the broadcast_data_set as a post. The Exposures entity is defined as a unique appearance, through a post, of a brand's sponsored content. Due to the fact a post can contain many pieces of branded content – even if they have the same asset and company – to avoid any redundancy, these pieces of data were separated. Finally, Data Snapshots are defined as a collection of user-interaction data – for a specific post and exposure – that is captured at multiple points in time.

Within the Posts entity you'll notice that pairs of channel and contenttype are separated. While they do have a partially transitive relationship – 3NF is still not violated but 4NF is – the many-to-many relationship they possess and combination limitations can cause data redundancy and integrity issues. For example, combination limitations are present in the fact that when channel = Youtube, the only contenttype possible is video. If we have the separated table, it's very easy to either modify the Content_Channel entity to include a new relationship or change the error in the pre-inserted data.

As a design decision, one could divide the Data Snapshot entity into separate tables; this division would depend on the original dataset each row came from. This would reduce storage, as there would not be as many null values, however it would increase the amount of joins needed during analysis. I have decided against this.

Part Two: Data Analysis

To get the data needed for the visualizations featured below I have used the data model and SQLite. The visualizations themselves are created using Python packages such as MatPlotLib and Seaborn. A powerpoint with the same information as below is also included in the deliverable. As an aside, in past research and internships the main method of data delivery would be interactive data webapps. This was handled through libraries such as Plotly Dash and StreamLit; they are extremely fast to develop.

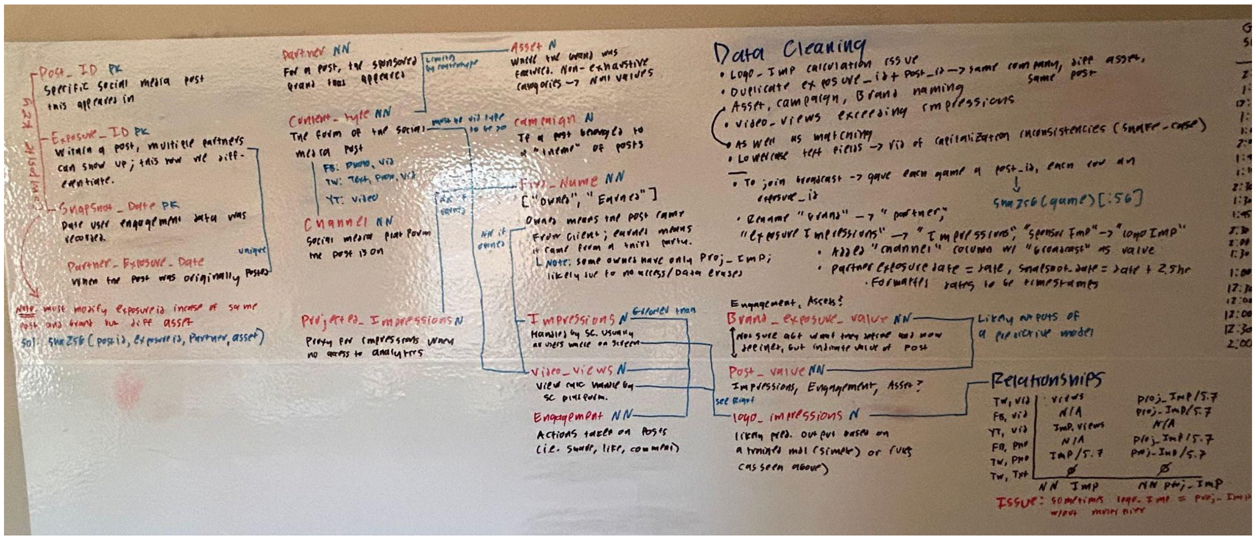
Before creating visualizations it is essential to define the *purpose* these visualizations are trying to accomplish. In the case itself this goal is stated as "evaluate their partnerships and value at the individual and collective partner level." I am defining my main goals as the (1) summarization of the combined datasets and (2) identification of important insights into this data. The insights created will be used to evaluate partnership value; hence, must be relevant to the client's business goals. Broadly, the business goals of a property within a sports league – concerning sponsor partnerships – is to offer value by creating visibility and exposure to their partners.

As a subjective choice, both organic and inorganic social media data will be included as "value-adding" within the following visualizations. While third-party creators would likely have posted content from these sports properties regardless of a brand taking a partnership deal, the brand itself would not be in these organic posts if it were not for the partnership.

Appendix

For the presentation I do not plan to exclusively use this document. I will create a dedicated slidedeck and only include the most important pieces of content. This slidedeck will aid the deliverable of Part Two. This document serves as an exhaustive look at everything I delved into for this case.

To better understand the social_dataset.csv and the interactions between columns, I mapped it out on my whiteboard. A more digestible version of this information is featured in part one.



Composite Key

Post_ID PK
Specific social media post this appeared in

Exposure_ID PK
Within a post, multiple partners can show up; this how we differentiate.

Snapshot_Date PK
Date user engagement data was recorded.

unique

Partner_Exposure_Date
When the post was originally posted

Note: must modify exposure id incase of same post and brand but diff asset
Sol: sha256(postid, exposureid, partner, asset)

Partner NN
For a post, the sponsored brand that appeared

Limited by content type

Asset N
Where the brand was featured. Non-exhaustive categories -> Null values

Content_type NN
The form of the social media post
FB: Photo, vid
TW: Text, Photo, Vid
YT: Video

must be vid type to be NN

Campaign N
If a post belonged to a "theme" of posts

Channel NN
Social media platform the post is on

NN if earned

NN if owned

Firm_Name NN
["Owned", "Earned"]
Owned means the post came from client; earned means it came from a third party.
L Note: some owned have only Proj_Imp; likely due to no access/Data erased

Projected_Impressions N
Proxy for impressions when no access to analytics

Impressions N
Handled by SC. Usually all users while on screen

Greater than

Video_views N
View calc handled by SC platform.

Engagement NN
Actions taken on posts (i.e. share, like, comment)

see Right

Data Cleaning

- Logo_Imp calculation issue
- Duplicate exposure_id + post_id -> same company, diff asset, same post

Asset, campaign, Brand naming

- video_views exceeding impressions
- as well as matching
- Lowercase text fields -> rid of capitalization inconsistencies (snake_case)

- To join broadcast -> gave each game a post_id, each row an exposure_id
  sha256(game)[:56]
- Rename "brand" -> "partner", "exposure Impressions" -> "Impressions", "sponsor Imp" -> "logo Imp"
  - Added "channel" column w/ "broadcast" as value
  - partner exposure date = date, snapshot_date = date + 2.5hr
    - Formatted dates to be timestamps

Engagement, Assets?

Brand_exposure_value NN
Not sure abt what they define and how defined, but indicate value of post

Likely outputs of a predictive model

Post_value NN
Impressions, Engagement, Asset?

logo_impressions N
likely pred. output based on a trained mdl (simple) or rules (as seen above)

Relationships

	NN Imp	NN proj_Imp
TW, vid	views	proj_Imp/5.7
FB, vid	N/A	proj_Imp/5.7
YT, vid	Imp, views	N/A
FB, Pho	N/A	proj_Imp/5.7
TW, Pho	Imp/5.7	proj_Imp/5.7
TW, Txt	∅	∅

ISSUE: sometimes logo_Imp = proj_Imp w/out multiplier

12:00
12:30
2:00