raw_data		
post_id	CHAR(7)	PK, FK
praw_tree	JSONB	

posts		
post_id	CHAR(7)	PK, FK
date	DATE	FK
subreddit	VARCHAR(21)	FK
upvote_score	INTEGER	
title	VARCHAR(300)	
selftext	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

posts_analysis		
post_id	CHAR(7)	PK, FK
post_sent_score	SMALLINT	
post_mh_score	SMALLINT	
total_sent_score	SMALLINT	
total_mh_score	SMALLINT	

responses		
response_id	VARCHAR(8)	PK, FK
post_id	CHAR(7)	FK
parent_id	VARCHAR(8) or NULL	FK
depth	SMALLINT	
upvote_score	INTEGER	
body	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

responses_analysis		
response_id	VARCHAR(8)	PK, FK
response_weight	SMALLINT	
sent_score	SMALLINT	
mh_score	SMALLINT	

subreddits		
id	SERIAL	PK
subreddit	VARCHAR(21)	FK
date	DATE	FK
sub_sent_score	SMALLINT	
sub_mh_score	SMALLINT	

raw_data		
post_id	CHAR(7)	PK, FK
praw_tree	JSONB	

posts		
post_id	CHAR(7)	PK, FK
date	DATE	FK
subreddit	VARCHAR(21)	FK
upvote_score	INTEGER	
title	VARCHAR(300)	
selftext	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

posts_analysis		
post_id	CHAR(7)	PK, FK
post_sent_score	SMALLINT	
post_mh_score	SMALLINT	
total_sent_score	SMALLINT	
total_mh_score	SMALLINT	

250 rows/day added
25 subreddits
Top 10 posts

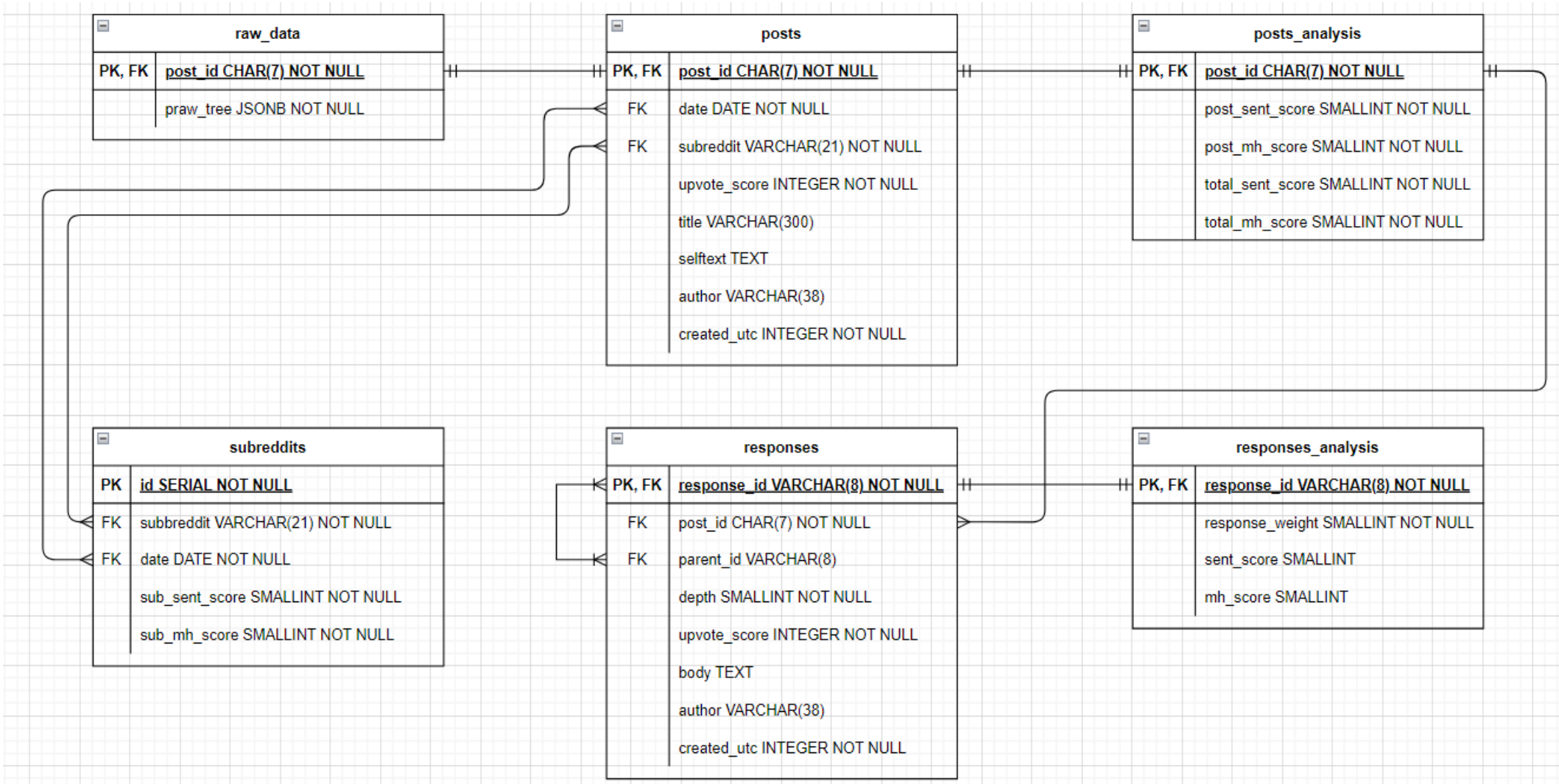
responses		
response_id	VARCHAR(8)	PK, FK
post_id	CHAR(7)	FK
parent_id	VARCHAR(8) or NULL	FK
depth	SMALLINT	
upvote_score	INTEGER	
body	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

responses_analysis		
response_id	VARCHAR(8)	PK, FK
response_weight	SMALLINT	
sent_score	SMALLINT	
mh_score	SMALLINT	

~25,000 rows/day
~100 rows per post based on tree depth and width limits

subreddits		
id	SERIAL	PK
subreddit	VARCHAR(21)	FK
date	DATE	FK
sub_sent_score	SMALLINT	
sub_mh_score	SMALLINT	

25 rows/day
1 row per subreddit

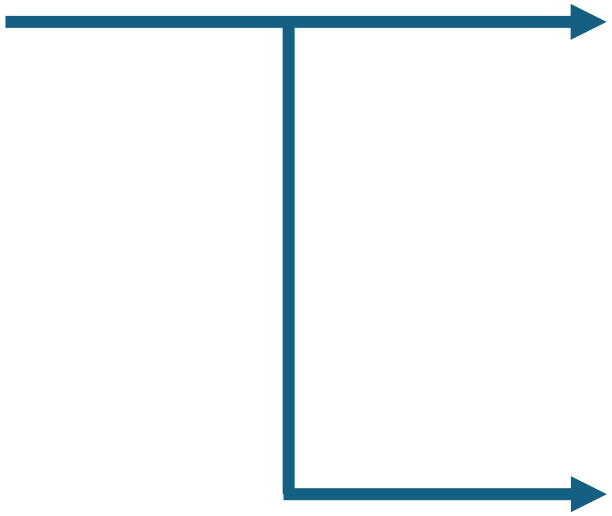

raw_data
PK, FK
post_id CHAR(7) NOT NULL
praw_tree JSONB NOT NULL
posts
PK, FK
post_id CHAR(7) NOT NULL
FK
date DATE NOT NULL
FK
subreddit VARCHAR(21) NOT NULL
upvote_score INTEGER NOT NULL
title VARCHAR(300)
selftext TEXT
author VARCHAR(38)
created_utc INTEGER NOT NULL
posts_analysis
PK, FK
post_id CHAR(7) NOT NULL
post_sent_score SMALLINT NOT NULL
post_mh_score SMALLINT NOT NULL
total_sent_score SMALLINT NOT NULL
total_mh_score SMALLINT NOT NULL
subreddits
PK
id SERIAL NOT NULL
FK
subbreddit VARCHAR(21) NOT NULL
FK
date DATE NOT NULL
sub_sent_score SMALLINT NOT NULL
sub_mh_score SMALLINT NOT NULL
responses
PK, FK
response_id VARCHAR(8) NOT NULL
FK
post_id CHAR(7) NOT NULL
FK
parent_id VARCHAR(8)
depth SMALLINT NOT NULL
upvote_score INTEGER NOT NULL
body TEXT
author VARCHAR(38)
created_utc INTEGER NOT NULL
responses_analysis
PK, FK
response_id VARCHAR(8) NOT NULL
response_weight SMALLINT NOT NULL
sent_score SMALLINT
mh_score SMALLINT

Step 1: Using Python, PRAW, and Orchestration, extract top 10 posts (limiting tree depth and width) from 25 pre-defined subreddits daily.

raw_data		
post_id	CHAR(7)	PK, FK
praw_tree	JSONB	

Step 2: Flatten JSONB into 2 tables:
1) Top-level posts
2) Responses (i.e., both comments and replies

raw_data		
post_id	CHAR(7)	PK, FK
praw_data	JSONB	



posts		
post_id	CHAR(7)	PK, FK
date	DATE	FK
subreddit	VARCHAR(21)	FK
upvote_score	INTEGER	
title	VARCHAR(300)	
selftext	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

responses		
response_id	VARCHAR(8)	PK, FK
post_id	CHAR(7)	FK
parent_id	VARCHAR(8) or NULL	FK
depth	SMALLINT	
upvote_score	INTEGER	
body	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

Step 3: Use upvote_score (relative* to the top-level post) and depth to calculate response_weight, which will later be used during post-level text-score aggregation in Step 5.

posts		
post_id	CHAR(7)	PK, FK
date	DATE	FK
subreddit	VARCHAR(21)	FK
upvote_score	INTEGER	
title	VARCHAR(300)	
selftext	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

responses		
response_id	VARCHAR(8)	PK, FK
post_id	CHAR(7)	FK
parent_id	VARCHAR(8) or NULL	FK
depth	SMALLINT	
upvote_score	INTEGER	
body	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

responses_analysis (partial)		
response_id	VARCHAR(8)	PK, FK
response_weight	SMALLINT	
sent_score	SMALLINT	
mh_score	SMALLINT	

Columns filled during step 3b

Columns yet to be filled (step 4)

* Relative to top-level post - Most likely using a scaling/dampening function such as log(n1)/log(n2). Different scaling functions explored.

Step 4: Process responses.body and CONCAT(posts.title, posts.selftext) through NLP, ML, and/or LLM to score text in multiple areas (sentiment, mental health, etc.)

posts		
post_id	CHAR(7)	PK, FK
date	DATE	FK
subreddit	VARCHAR(21)	FK
upvote_score	INTEGER	
title	VARCHAR(300)	
selftext	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

posts_analysis (partial)		
post_id	CHAR(7)	PK, FK
post_sent_score	SMALLINT	
post_mh_score	SMALLINT	
total_sent_score	SMALLINT	
total_mh_score	SMALLINT	

Columns filled during step 4

Columns yet to be filled (step 5)

responses		
response_id	VARCHAR(8)	PK, FK
post_id	CHAR(7)	FK
parent_id	VARCHAR(8) or NULL	FK
depth	SMALLINT	
upvote_score	INTEGER	
body	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

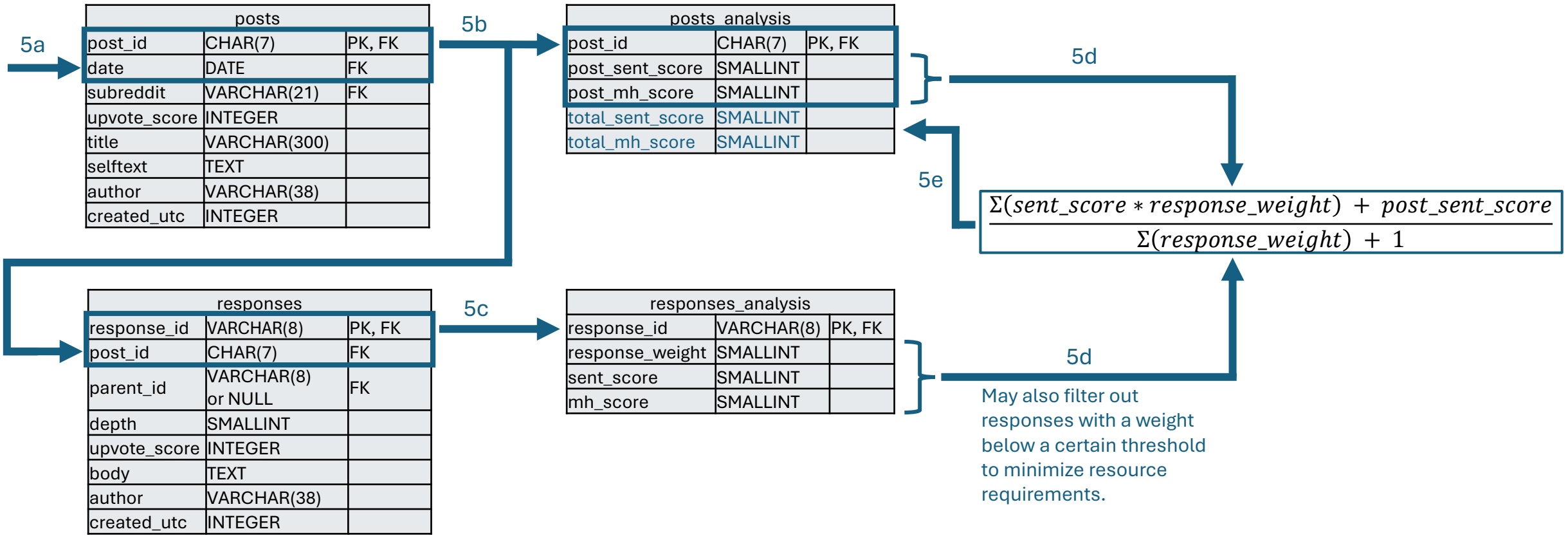
responses_analysis		
response_id	VARCHAR(8)	PK, FK
response_weight	SMALLINT	
sent_score	SMALLINT	
mh_score	SMALLINT	

Columns already filled from step 3

Columns filled during step 4

Step 5: Post-level text-score aggregation.
Aggregate all submission text scores (i.e., for all posts, comments, and replies) into a total score for each post.

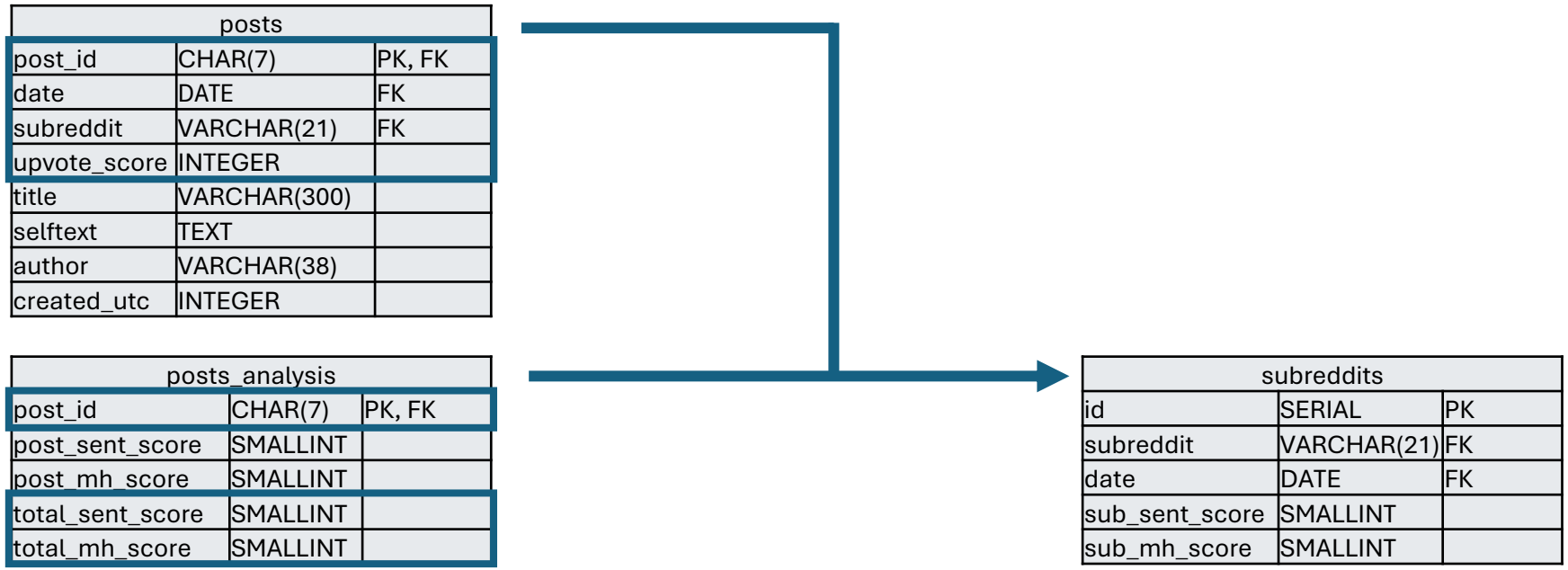
a. Retrieve all posts from the “posts” table that match the specified date.
b. Get all responses from the “responses” table that belong to these posts.
c. Join the “responses” with their analysis data from the “responses_analysis” table.
d. Calculate the weighted average text scores for each post using aggregation.
e. Update the “posts_analysis” table with the calculated total text scores for each post.



Step 6: Subreddit-level text-score aggregation.
Aggregate all post text scores into a total score for each subreddit for that day, using upvote_score to weight posts relatively*

a. Retrieve all posts from the “posts” table that match the specified date.
b. Calculate post weights based on upvote scores and join with sentiment scores.
c. Group posts by subreddit and date, then compute daily subreddit text-scores.
d. Insert these text-scores for the given date into the subreddits table.

* Relative to top post for that subreddit for that day - Most likely using a scaling/dampening function such as log(n1)/log(n2). Different scaling functions explored.



posts		
post_id	CHAR(7)	PK, FK
date	DATE	FK
subreddit	VARCHAR(21)	FK
upvote_score	INTEGER	
title	VARCHAR(300)	
selftext	TEXT	
author	VARCHAR(38)	
created_utc	INTEGER	

posts_analysis		
post_id	CHAR(7)	PK, FK
post_sent_score	SMALLINT	
post_mh_score	SMALLINT	
total_sent_score	SMALLINT	
total_mh_score	SMALLINT	

subreddits		
id	SERIAL	PK
subreddit	VARCHAR(21)	FK
date	DATE	FK
sub_sent_score	SMALLINT	
sub_mh_score	SMALLINT	

Results “subreddits” table: Data available for plotting, dashboard connection, trend analysis, csv export, etc.

subreddits		
id	SERIAL	PK
subreddit	VARCHAR(21)	FK
date	DATE	FK
sub_sent_score	SMALLINT	
sub_mh_score	SMALLINT