SIADS 593: Fantasy Football Rookie Running Back Success Analysis

Grant Jason, Jeremy Tang, Zain-Ul-Abidin Adhami

Background & Motivation

Background

Every year, over 50 million people sign up for and participate in a fantasy football league, with that total growing nearly 7% every year[1]. Fantasy football is a game where participants draft gridiron football players to their teams, and each player's real world performance impacts scoring for the fantasy football leagues.

Fantasy leagues are maintained among friends every year, with some leagues having over a decade's worth of history. Winning your fantasy league is a huge point of pride, with plenty of bragging rights that accompany it. On the flipside, some leagues impose hilarious punishments for finishing at the bottom.

Motivation

Our team decided to do a project that would generate some unique insights on NFL Rookie performance, as they are frequently either hidden gems (Alvin Kamara in 2017) or huge "busts" that fail to live up to their hype (Marvin Harrison Jr. in 2024).

Intent

Specifically, our team decided to focus on rookie running backs. Running backs can have the biggest impact on your fantasy team, and the window for a running back in the real world to perform at their peak is very small (No running back over the age of 29 has ever finished as fantasy RB1). As such, NFL teams frequently lean on the youth and talents of rookie running backs. Without a history of playing in the league, how can you assess whether a rookie will be a good fantasy football player?

Our team hopes that this project can uncover some unique insights on rookie running back performance, and potentially use it in future fantasy leagues to identify that year's diamond in the rough, or confidently pass on that year's hyped up rookie sensation.

Questions

[1] Are the running back's college stats a potential indicator of "fantasy" success? (Interpret as: Top 10 scoring in their position) [2] Does where a running back was taken in their draft impact fantasy scoring? [3] How much do physical attributes (i.e. NFL Combine measurements) matter for fantasy football success?

1. https://thefsga.org/industry-demographics/

Data Sources

Primary Dataset: NFL Player-Level Data (nflverse / nfl_data_py)

- **Where is it located?**: nfl_data_py (nflverse) - a public Python package + GitHub docs.
  - Link: https://github.com/nflverse/nfl_data_py
  - Sub-imports used: import_players, import_draft_picks, import_combine_data, import_seasonal_data.
- **Access Method**: pip install nfl_data_py → import nfl_data_py as nfl → use endpoints like import_weekly_data(years=[...])
- **Formats returned/used:** Loaded directly into **pandas DataFrames** (underlying sources are CSV/JSON maintained by nflverse). Our group exported the tables to CSVs as needed.
- **Important Variables:**
  - **Players**: gsis_id, display_name, pfr_id, position, college_name, rookie_season.
  - **Draft Picks (2015-2025)**: season, round, pick, team, pfr_player_id, pfr_player_name, college, position.
  - **Combine Stats (2015-2025)**: ht, wt, forty, bench, vertical, broad_jump, cone, shuttle.
  - **Seasonal Stats (2015-2024)**: carries, rushing_yards, rushing_tds, rushing_fumbles, rushing_fumbles_lost, receptions, targets, receiving_yards, receiving_tds, receiving_fumbles, receiving_fumbles_lost.
- **How many records were retrieved?**
  - **Seasonal Stats**: full league coverage for **2015-2024** regular seasons (**~ 35k-40k rows** per season).
  - **Draft Picks**: All Draft Picks between **2015-2025** (one row per pick; filtered to **RBs**).
  - **Combine Stats**: The full Combine Cohort between **2015-2025** (**~ 5,062 rows**).
  - **Players**: all players with rookie_season ≥ 2015, filtered to **RBs**.
- **Time Periods Covered**:
  - Draft + Combine: **2015-2025**
  - NFL Seasons: **2015-2024 (Regular Season),** filtered to **RBs.**

Secondary Dataset: College Football Player Stats (CFBD API)

- **Where is it located?**: College Football Data API (CFBD).
  - Link: https://api.collegefootballdata.com/
- **Access Method**: pip install cfbd → set API key → call player/season stats.
- **Formats returned/used**: **JSON** responses → normalized to **pandas DataFrames**. Our group exported the tables to CSVs as needed.
- **Important Variables:**
  - Raw fields: season, playerId, player, position, team, conference, category, statType, stat.
  - We filtered category = "rushing" and pivoted statType → CAR, TD, YDS, LONG, then renamed to college_carries, college_tds, college_yards, college_long and aggregated to career-level per player (keeping latest college/team).
- **How many records were retrieved?**: **1,292,723 rows** across the seasons pulled (concatenated df_all in the notebook).
- **Time periods covered**: **2010-2025**
  - Our group used a wide window, so rookies from **2015-2025** have complete prior college history).

Data manipulation – initial retrieval

Overview

Our project has 2 notebooks. The first, ff_data_ingestion_cleaning.ipynb does all the heavy lifting to retrieve the data, clean it, and manipulate it. The notebook outbooks the final data frame to a .csv which is then read by the other notebook, ff_analysis_visualization.ipynb. We did this so the user doesn't need to re-run any of the computationally intense ingestion code blocks, and so the final dataframe is stable in a csv form.

Task #1: NFL Draft Results from 2015-2025

To start, we use the `import_draft_picks()` function available to us within the nfl_data_py package. The function takes a list of years and returns the draft data from those years. Although it appears to be 10 years, it is actually 11 unique drafts, so we pass in `range(2015, 2026)` to the function to generate a list of the 11 years we'll be pulling drafts for.

The dataset is relatively straightforward and already pretty clean. To make it a less computationally expensive later on, we filter the dataframe using the overload method to pull in just the columns we think we will need.

Task #2: Combine Results from 2015-2025

Similar to step one, we can call the `import_combine_data()` function to bring in combine results from nfl_data_py. This dataframe is much smaller, but provides us key information for a running back like their 40 yard dash time, cone, and shuttle drills. We also notice that the combine results has the same `pfr_id` as the draft results. This will provide us a straightforward unique key to use, as this key appears to be generated per distinct player. It will help us avoid some of the scenarios like Josh Allen the QB, and Josh Allen the DE.

Task #3: College Football Stats from 2010-2025

To call this API, we use the requests library and pass in our API key. The response is in JSON, so we then use the `json_normalize()` function in pandas to load it as a dataframe.

One unique thing about the API is we need to designate the year we are going to pull before making the API call. Since we want to retrieve data for a bunch of years, the best way to do this is via a for loop over a list of the years, and pass in the year via an f string to the URL string we're giving to `requests.get()`

Once we retrieve the data for that year, we append it to a list which we then concatenate together into one dataframe.

Data manipulation - merging and cleaning pt 1

Task #4: NFL Player Rookie Season Stats 2015-2025

Finally, we want to grab the full season stats for how every rookie performed in their first year. The `import_seasonal_data()` function takes a list of years, and a season time period designation (preseason, regular, post season). Since regular season is the only relevant one for fantasy, we'll call that out specifically.

The seasonal data return does not include the pfr_id which we were planning to use to join the draft and combine data earlier. So we also call `import_players()` which has the "gsis_id" column that are shared between these two tables, as well as the `pfr_id` we needed. We call `.merge()` to left join the seasonal data to the player data, and this gives us our final table we will use in the analysis.

Task #5: Filter Draft Data and Apply Position Rankings

To focus our analysis, we first filtered the dataset to only include running backs. We then calculated a **positional rank** because a player's overall draft pick can be misleading when comparing across years. For example, the first running back selected might be a top-10 pick in a strong draft class but a 2nd-round pick in a weaker one. A positional rank (e.g., RB1, RB2) normalizes this by showing the order in which running backs were taken relative to their peers *within the same draft*. To achieve this, we sorted the data by `season` and `pick`, then using a `groupby()` on the `season` column and applying a cumulative count function. This restarts the count for each new draft class. During this process, we also identified and dropped a single record with a null `pfr_id` to ensure data integrity for future joins.

Task #6: Merge Draft and Combine Data

Next, we want to merge the combine data and the draft data. We're going to use the draft data as a starting point and always left join to it, since this effectively "filters" our following datasets and only pulls in relevant matches from them. To start, we'll identify the relevant columns from the combine data we want to pull using a list. Then, we use `merge()` and join the combine data to the draft data on the `pfr_id.` After the merge, we drop the duplicate pfr_id column from the combine data.

Later on, we need to merge this dataset with college data. There is no shared identifier between them, so we'll create one using the player's full name, position, and the college they attended. To do this, we'll combine the 3 columns where that data is located. To make the join key cleaner, we'll first apply `lower()` and then `replace()` using a regex to identify and strip any "special" characters (Such as the "." or "-" in some player names). The regex expression we use is `r'[^a-z0-9]'` which captures all non lowercase or numerical values. We don't worry about leaving upper case letters out of this since we applied `lower().`

Data manipulation - merging and cleaning pt 2

Task #7: Filter and Aggregate College Data

Now, we want to bring in their college career stats. First, we'll filter the dataframe to only the rushing category. Next, we noticed there were a few instances where a known running back (Sony Michele from University of Georgia) had a "?" as their position, which caused the merge later on to fail. Since we've filtered to just rushing data, and we'll be joining this against a list of only Running Backs in the NFL draft, we use a `str.replace()` to swap all of the ? with RB to ensure our unique key generation later on succeeds. Next, we have to pivot and aggregate the data since the college data to a player-season-position-college level since this will be used later on. We remap the name of the stat columns to include "college" so we don't have two "rushing yards" columns later on when we add in professional stats. Finally, another thing we noticed was a lot of the numeric values were actually stored as strings, which led them to concatenate the strings together instead of adding the numeric values. To remedy this, we use `.apply(pd.to_numeric)` to cast the relevant columns to numbers.

We then needed to aggregate the season level stats to a career level stat. First, we sort all values by playerID and season. We do this because inside our `groupby()` we actually need to identify what year was the last a college player played, and what school they attended in that year. If we're going to use name-position-school as a unique identifier, then we need only the last school they played at since this is what is frequently cited as the school they attended during the NFL Draft. For all other categories, we sum the values. Now that we have aggregated, we do another sanity check and notice there were actually handful of duplicates in our unique identifier. This is because previously in our pivot, we used playerID as one of the key columns, and as it turned out, there were a few instances where the same player's name was slightly different in the data, which led to the API generating two different playerIDs. To remedy this, we perform one more aggregating, this time against our unique identifier. Our remaining dataframe now has the unique identifier we need to join to the draft data, and columns for all of their college stats.

Task #8: Merge College & NFL Rookie Season Data

We have 3 tables now to merge together - The draft + combine data, the college data, and the rookie season data. Thankfully, all of the preparation we did beforehand makes this straightforward. First, we left join the college data to the draft + combine data on the uniqueID. Next, we left join the rookie season stats to the draft + combine + college stats on the `pfr_id`. And voila! We have our full dataset. The final step is to calculate fantasy points.

Task #9: Calculate Full Season PPR Fantasy Score

Finally, with all of our data joined and cleaned, we can calculate a player's fantasy score for their rookie season. To do this, we define a custom function `calculate_fantasy_points()` which takes a single argument we're calling row. The intention is to use this function in an `apply()` method on the data frame and save the results in a few column. Since we're using apply, we know the function is applied to each row of the dataframe. Because of this, we can just access the column we need to calculate the score directly now within the function. Doing this, we apply Points per Reception (PPR) scoring to the full season NFL stats we have. Before calling the function, we replace all NaN values with a 0 in the applicable stat columns so the calculation succeeds. Finally, we call the `apply()` function and pass in our `calculate_fantasy_points()`, saving the results in a new column in the dataframe.

In the final code block, we use the pathlib library to dynamically identify the home folder path on the user's computer. We then add 'Downloads' to the file path, then pass that entire path to into `to_csv()` function to download the final dataframe as a csv.

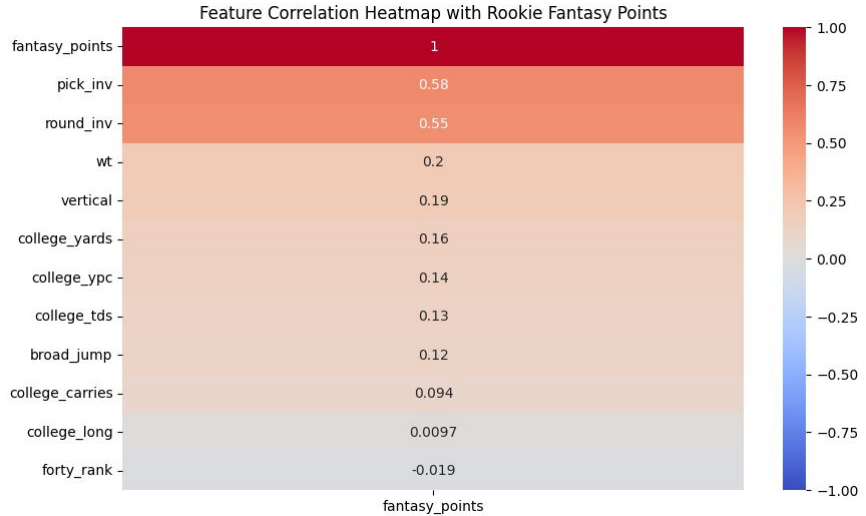
Analysis

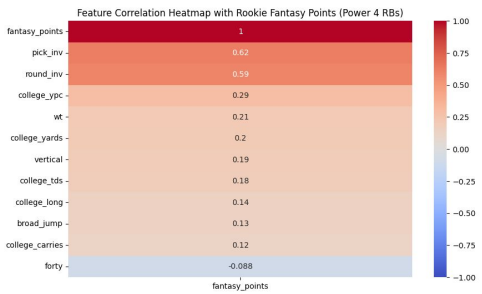
Our analysis and visualization section is in a separate notebook, we take the output csv file from the ingestion and manipulation sections and read it as the input.

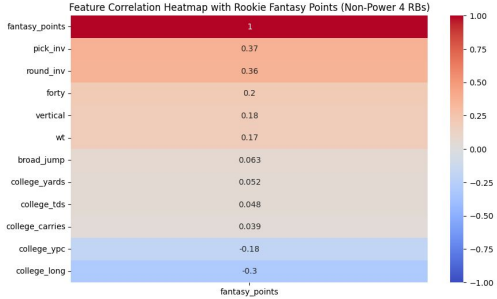
Our first step was to identify which attributes from the combined dataset are historically most correlated with rookie-year production, using a correlation heatmap as a baseline analysis. A heatmap is good in this scenario because it shows correlation across all the quantitative attributes to the target variable. We selected this color scheme from red (hot) to blue (cold) because it is naturally expressive showing relative correlation strength naturally. We selected numeric columns and removed variables with more than 50% missing values (cone, shuttle, and bench). We 'flipped' the round, pick, and forty time variables, since lower values in these features actually indicate better performance. This ensures their correlations with rookie fantasy points are positive. A draft round of 1 is better than round 3, so smaller numbers represent stronger performance. The resulting heatmap is shown on the top right.

- **Draft pick and round order show the highest correlation** to a running back's rookie fantasy points. This makes sense in the real world as typically draft position is a summary signal of an NFL team's evaluation of a player's talent and opportunity.
- **Weight (wt) and vertical combine measurements are the highest correlated combine features** with college yards and ypc as the highest college stat features.
- Surprisingly **forty_rank is the least correlated** with fantasy points. This is surprising because the 40 Yard Dash is often the most popular combine measurement that NFL fans follow. One would think that a RB's straight line speed would help indicate strong performance, but the heatmap shows otherwise

We then decided to take a look at the same heatmaps but **splitting between RBs from Power 4 and non Power 4 schools.** Observing the resulting two different heatmaps, we notice some differing things. First we see **a stronger correlation in the pick/round selection for Power 4 schools (0.62 and 0.59) than non-Power 4 (0.37 and 0.36).** It iis very common for P4 RBs to be selected earlier and have more success coming from larger schools with media exposure, stronger college competition, and higher talent evaluations. Non-P4 RBs typically get selected later, so the pick order does not correlate as highly to success since NFL scouts may miss talent from smaller schools and the variance is much higher. It is also interesting to see that **college stats are more correlated for P4 players and very little correlated for non-P4**. This could be because in non-P4 games, there are many opportunities for RBs to "stat pad" against lesser competition, so with higher variance in smaller schools could cause it to be less correlated with NFL rookie year success.





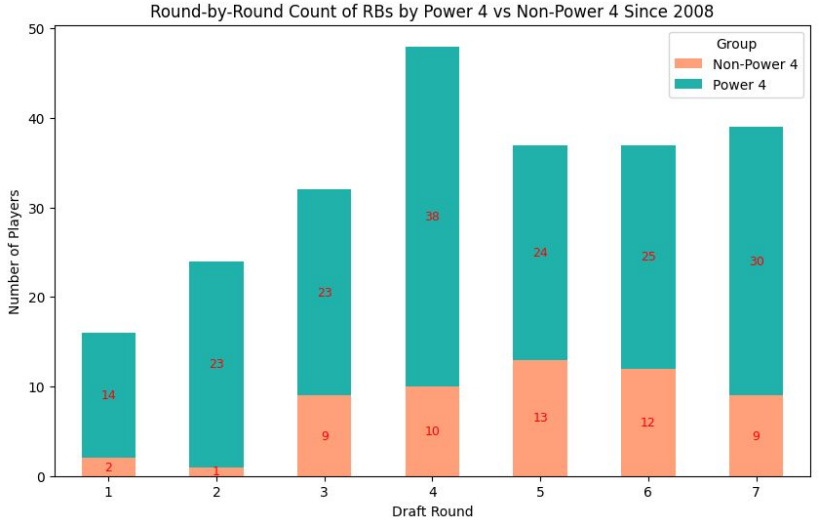


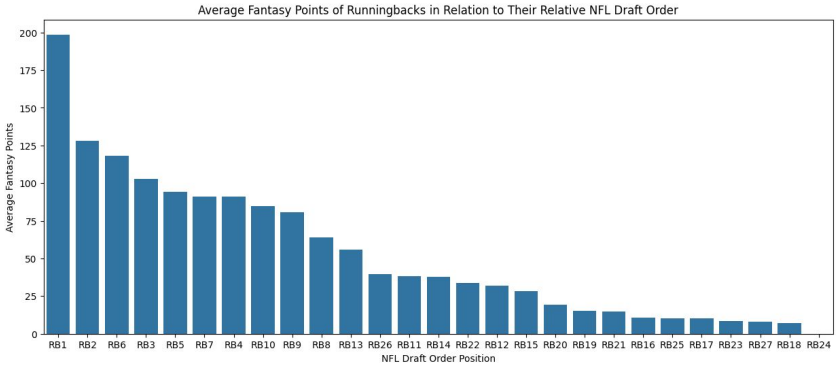
Analysis cont

We confirmed our hypothesis that non-Power 4 running backs tend to be drafted later, resulting in a weaker correlation between draft position and rookie performance. A stacked bar chart of RB selections by round shows that only three non-P4 RBs were taken in the first two rounds, while a larger number are drafted in the third round or later. This distribution helps explain the greater variability and lower correlation for non-P4 RBs. The pattern likely reflects that **teams devote less scouting and resources to smaller schools, making it statistically harder to identify strong late-round performers.** A stacked bar chart is effective in this scenario because it not only effectively shows the total number of RBs by round, but the splitting of the bar by P4 vs non-P4 shows the differentiation. We selected a green and orange color because they contrasted against each other well and clearly expressed the difference in each bar.

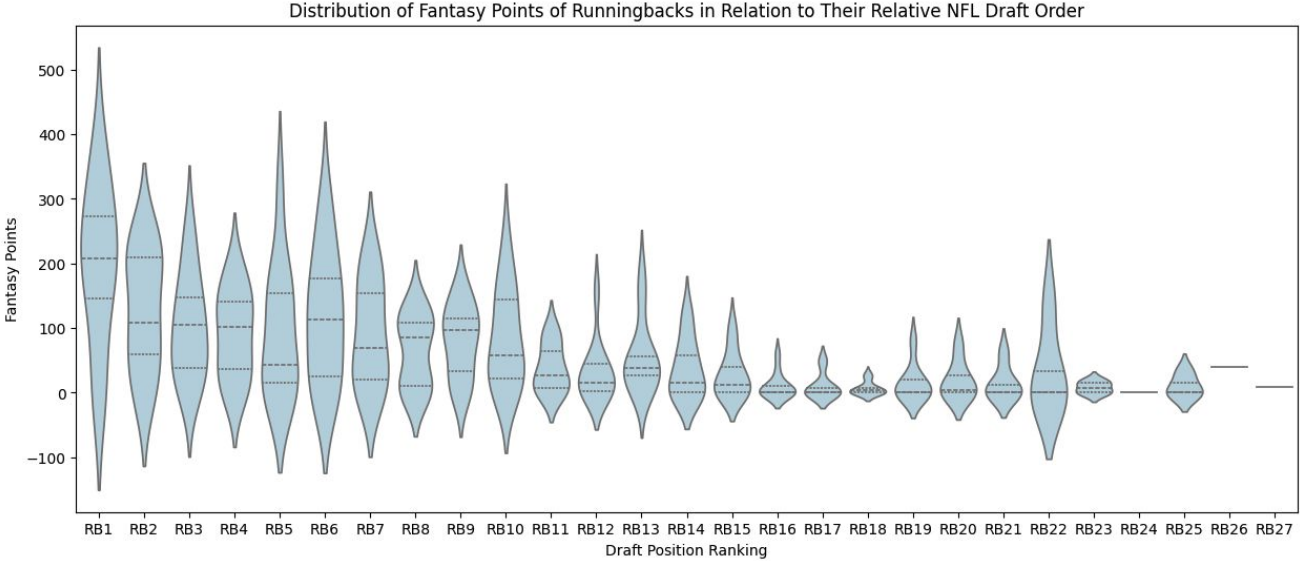
Focusing back on the top two highest correlated features to the target variable, we wanted to plot the average fantasy points of RBs based on their “position_rank”. Position rank is each RB’s relative draft position within their class. For example, the 'RB1' refers to the first running back selected in a given draft. In 2023, that was Bijan Robinson (Round 1, Pick 8), while in 2024 it was Jonathan Brooks (Round 2, Pick 46). We do this using a seaborn barplot along with groupby functions on the df on the position_rank column.

While it may seem obvious, the first running back drafted (RB1) generally delivers the strongest rookie season. What’s more interesting, however, is what happens after RB1: **there’s a sharp drop in average performance, followed by a leveling off after RB3 that continues until roughly RB10**. This pattern suggests that, in fantasy drafts, running backs outside the top two should often be viewed as roughly interchangeable **making it less worthwhile to 'reach' for a specific rookie RB based purely on hype.** We decided to take this one level deeper and see if there were any interesting trends in the distributions within each position rank.





Analysis cont



In the violin version of the graph, we get some additional insights. A violin plot is more expressive than a bar chart showing distribution rather than just average. It lets us see if performance is skewed and if it has a few outliers. It is more effective in showing variation at each relative position. Although the mean for each position after 1 was pretty consistent, the quartiles and extremes are much different. **The biggest surprise is seeing how much higher RB2's upper quartile (~200 fantasy points) is compared to RB3 (~150).** As expected, this suggests that if you're going to pick a rookie running back, you're better off putting emphasis on picking running backs that were picked as the 1st or 2nd running back in their draft. This information gives us some strong indicators of where a rookie running backs draft position should truly lie in fantasy drafts. Rookies as we mentioned can sometimes be extremely hyped, with Ashton Jeanty going #9 overall by ADP this year. By all of our indicators, he should perform quite strong in fantasy (RB1, 6th overall in his NFL Draft). Based on our analysis, he is likely to score on average 200 points his rookie season, with an upper quartile of ~275. At this amount, he would finish somewhere between RB10 and RB21 in 2024[1]. Based on this, his ADP should range somewhere from 17th to 47th overall. However, in 2025, Ashton Jeanty on average was being drafted as RB6 with an overall ADP of 9[2]. Based on our analysis, we would say Jeanty was likely being overdrafted at this position.

1. https://fantasydata.com/nfl/fantasy-football-leaders?scope=season&sp=2024_REG&position=rb&scoring=fpts_ppr&order_by=fpts_ppr&sort_dir=desc

2. https://www.fantasypros.com/nfl/adp/rb.php

Statement of work

Zain-Ul-Abidin Adhami

Zain set up the ingestion notebook, handled the pip installs, created the API key for the CFB data set, wrote code for all of the data ingestion and discovery, and helped with some of the analysis work including the P4 v Non P4 level of detail

Grant Jason

Grant filled out the ingestion notebook with cleaning and manipulation steps, documentation, markdown formatting of the notebook, and creation of the project report theme. Grant also helped created the position_rank metric and analysis.

Jeremy Tang

Jeremy helped set up the team's github page, led the bulk of the project's analysis such as the correlation heatmaps, and write up of the findings from all the analysis.

Collaboration

To collaborate on the project, our team did a weekly check in with one another and stayed in regular contact over an iMessage group chat. If we were to recommend a way to improve collaboration in the future, having more live working sessions together would be beneficial to help talk through some of our analysis ideas before writing code.