Part 1: Foundational Data Analysis

Kickstarter, the crowdfunding platform, is thinking about providing a consulting service to project founders to help its customers create more successful crowdfunding campaigns. You've been asked to do some initial analysis.

1. In order to complete this assignment, please follow the link and download the following Kickstarter dataset: http://bit.ly/2cgMGDm. Use the dataset to complete this task.

We are tools agnostics, you can use any tools you'd like: Excel, Python, SQL, Tableau, Google spreadsheets, etc. We recommend using whatever tool you know best, we're here to find out how you think about solving data problems, not which tool you can do it in.

2. **What is the mean (total) pledge that projects get**? (not per backer)
Your answer may vary by +/- 5%

3. **Create a histogram that shows the distribution for number of backers. What is the skew of the distribution?**

4. **Is the 'duration' variable normally distributed?**

5. **If you could collect data on another attribute of these projects, what would it be and why?**

1. Preparing data for analysis

1. Loading data.

To prepare the data for analysis, I loaded the dataset into an Oracle12c database, utilizing an external table:

```
create table dsi_kickstarterscrape_ds
(
     project_id               number,
     name                     nvarchar2(500),
     url                      varchar2(200),
     category                 varchar2(100),
     subcategory              varchar2(100),
     location                 nvarchar2(100),
     status                   varchar2(50),
     goal                     number,
     pledged                  number,
     funded_percentage        number,
     backers                  number,
     funded_date              varchar2(50),
     levels                   number,
     reward_levels            varchar2(1000),
     updates                  number,
     comments                 number,
     duration                 number
)
organization external
(
  type oracle_loader
  default directory csv_files
  access parameters
  (
      records field names all files
      characterset we8mswin1252
      fields terminated by ','
      optionally enclosed by '"' and '"'
      missing field values are null
  )
  location
  (
    'dsi_kickstarterscrape_dataset.csv'
  )
)
/

CREATE TABLE dsi_kickstarterscrape_ds_i AS
SELECT * FROM dsi_kickstarterscrape_ds;


SELECT COUNT(*) FROM dsi_kickstarterscrape_ds_i;

Count: 45957
```

2. Discovering duplicate rows in the dataset.

When I attempted to create the primary key for this table, Oracle found duplicate rows in the dataset:

```
ALTER TABLE dsi_kickstarterscrape_ds_i
ADD CONSTRAINT pk_project_id
PRIMARY KEY (project_id);
```

```
Error report -
SQL Error: ORA-02437: cannot validate (C##DEV.PK_PROJECT_ID) - primary key violated
02437. 00000 -  "cannot validate (%s.%s) - primary key violated"
*Cause:    attempted to validate a primary key with duplicate values or null values.
*Action:   remove the duplicates and null values before enabling a primary key.
```

The following query returned **142 rows** containing PROJECT_ID values of the duplicate rows:

```
SELECT project_id, COUNT(*)
  FROM dsi_kickstarterscrape_ds_i
 GROUP BY project_id
HAVING COUNT(*) > 1;
```

3. Identifying and removing duplicate rows.

I used the following query to identify all duplicate rows in the dataset:

```
SELECT * FROM dsi_kickstarterscrape_ds_i
 WHERE project_id IN
       (SELECT project_id
          FROM dsi_kickstarterscrape_ds_i
         GROUP BY project_id
        HAVING COUNT(*) > 1)
 ORDER BY project_id;
```

I compared duplicate rows – in the Oracle table and in the original Kickstarter dataset – and found them 100% identical.

19820	506764408	iMAGINARY wORLDS	http://www.kickstarter.com/projects/391021092/imaginary-worlds	Publishing
19821	1727894974	IMAGINATION HEAD NEW ALBUM	http://www.kickstarter.com/projects/294797067/imagination-head-new-album	Music
19822	281344017	Imagination Situation - A Webseries For Parents And Kids	http://www.kickstarter.com/projects/1102654676/imagination-situation-a-web-series-for-parents-and	Film & Video
19823	603454766	Imagine Me	http://www.kickstarter.com/projects/692653576/imagine-me	Games
19824	1051737169	Imagine! Dream! Explore! Wheel of the Year Children's Books	http://www.kickstarter.com/projects/1462174029/imagine-dream-explore-solstice-moon-solstice-sun	Publishing
19825	191126140	Imagine: A reader active comic book co.!	http://www.kickstarter.com/projects/1042659802/imagine-a-reader-active-comic-book-co	Comics
19826	2140333236	Imagined Family Heirlooms: An Archive of Inherited Fictions	http://www.kickstarter.com/projects/andersonstaley/imagined-family-heirlooms-an-archive-of-inherited	Photography
19827	2140333236	Imagined Family Heirlooms: An Archive of Inherited Fictions	http://www.kickstarter.com/projects/andersonstaley/imagined-family-heirlooms-an-archive-of-inherited	Photography
19828	1984248049	Imagining Shakespeare	http://www.kickstarter.com/projects/kevinsprague/imagining-shakespeare	Publishing
19829	648882926	imAnerd Clothing: It's a new NerdStyle Brand	http://www.kickstarter.com/projects/1017911160/a-more-perfect-union	Fashion
19830	2133686007	Imani Uzuri "The Gypsy Diaries" Kickstarter Album Campaign!	http://www.kickstarter.com/projects/1839889570/imani-uzuri-the-gypsy-diaries-kickstarter-album-ca	Music
19831	674235454	Imani Worship - Nashville Bound	http://www.kickstarter.com/projects/1217380520/imani-worship-nashville-bound	Music
19832	1057374918	iMasterCleanse	http://www.kickstarter.com/projects/mikeolaski/imastercleanse	Film & Video
19833	1311084698	IMBC Fashion	http://www.kickstarter.com/projects/jamesd31/imbc-fashion	Design
19834	913297740	IMDB Credit & Film Course From Diabolical Dave?	http://www.kickstarter.com/projects/syberfilm/imdb-credit-and-film-course-from-diabolical-dave	Film & Video
19835	833441893	iMEE... reconceiving the art of dance theatre	http://www.kickstarter.com/projects/1177956491/imee-reconceiving-the-art-of-dance-theatre	Dance

The duplicate rows have been removed from the table:

```
DELETE FROM dsi_kickstarterscrape_ds_i d
 WHERE d.ROWID IN
       (SELECT a.ROWID AS rid
          FROM dsi_kickstarterscrape_ds_i a
         MINUS
        SELECT MIN(ROWID) AS rid
          FROM dsi_kickstarterscrape_ds_i kp
         GROUP BY project_id)
```

```
142 rows deleted.
```

For data analysis I used **SQL**, **PL/SQL** and a built-in Oracle package **DBMS_STAT_FUNCS.**

2. What is the mean (total) pledge that projects get? (not per backer)

Your answer may vary by +/- 5%

Answer: The mean (total) pledge is 4980.75 *

---

1. Calculating mean for the column PLEDGE:

```
SELECT ROUND(SUM(pledged)/COUNT(pledged), 2) AS mean
FROM dsi_kickstarterscrape_ds_i;
```

Mean: 4985.74

or

```
SELECT ROUND(AVG(pledged), 2) AS mean
FROM dsi_kickstarterscrape_ds_i;
```

Mean: 4985.74

* For the original dataset (with duplicate rows), both methods produced **4980.75.**

3. Create a histogram that shows the distribution for number of backers. What is the skew of the distribution?

Answer: Skewness = 87.34 (which indicates a highly skewed distribution). The curve is positively skewed.

---

1. Collecting statistical data for the column BACKERS, using procedure **SUMMARY** from the built-in Oracle package **DBMS_STAT_FUNCS**.

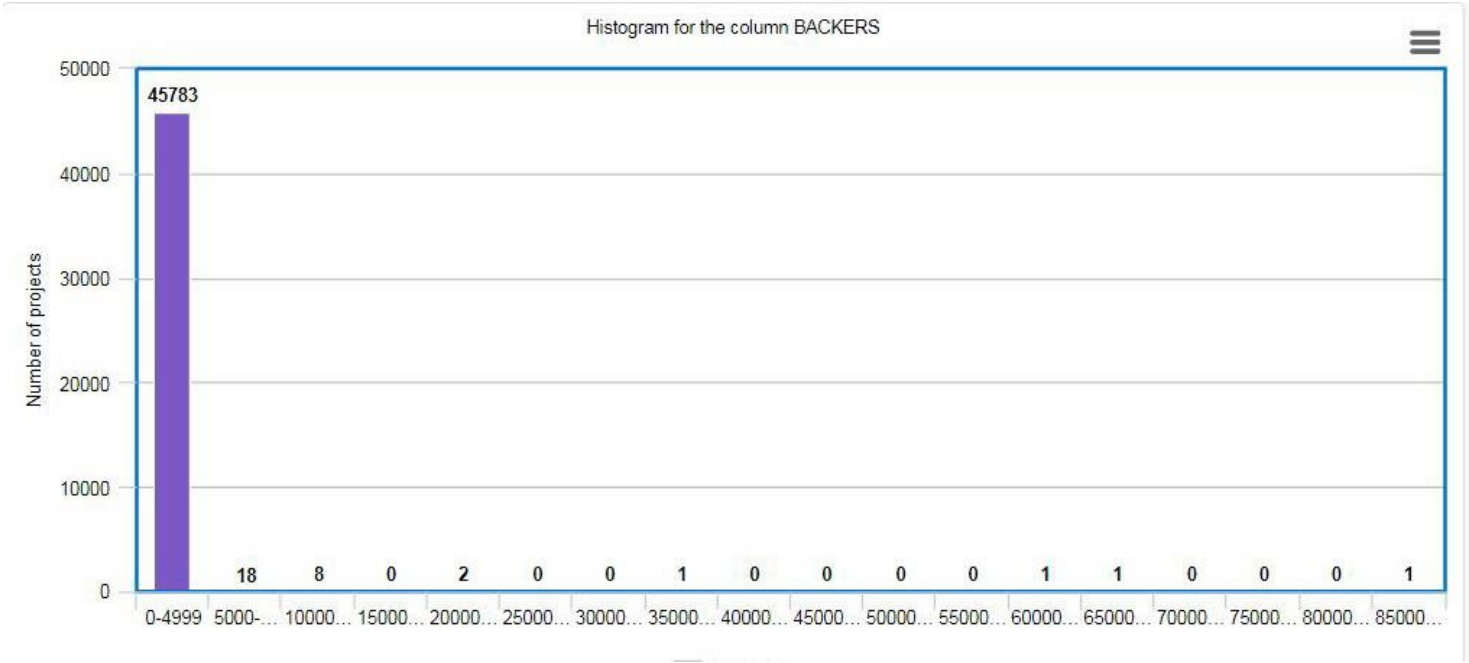
```
DECLARE
   summary_type_out dbms_stat_funcs.SummaryType;
BEGIN
   dbms_stat_funcs.summary('C##DEV', 'DSI_KICKSTARTERSCRAPE_DS_I', 'BACKERS', 3, summary_type_out);

   dbms_output.put_line('Summary statistics for column BACKERS in the table ' ||
                                                              'DSI_KICKSTARTERSCRAPE_DS_I');
   dbms_output.put_line('Count: '        || summary_type_out.count);
   dbms_output.put_line('Min: '          || summary_type_out.min);
   dbms_output.put_line('Max: '          || summary_type_out.max);
   dbms_output.put_line('Range: '        || summary_type_out.range);
   dbms_output.put_line('Mean: '         || round(summary_type_out.mean));
   dbms_output.put_line('Mode Count: '   || summary_type_out.cmode.count);
   dbms_output.put_line('Mode: '         || summary_type_out.cmode(1));
   dbms_output.put_line('Variance: '     || round(summary_type_out.variance));
   dbms_output.put_line('Stddev: '       || round(summary_type_out.stddev));
   dbms_output.put_line('Quantile 25: '  || summary_type_out.quantile_25);
   dbms_output.put_line('Median: '       || summary_type_out.median);
   dbms_output.put_line('Quantile 75: '  || summary_type_out.quantile_75);
   dbms_output.put_line('Top 5 values: ' || summary_type_out.top_5_values(1) || '-' ||
                                            summary_type_out.top_5_values(2) || '-' ||
                                            summary_type_out.top_5_values(3) || '-' ||
                                            summary_type_out.top_5_values(4) || '-' ||
                                            summary_type_out.top_5_values(5));
   dbms_output.put_line('Bottom 5 values: ' || summary_type_out.bottom_5_values(5) || '-' ||
                                               summary_type_out.bottom_5_values(4) || '-' ||
                                               summary_type_out.bottom_5_values(3) || '-' ||
                                               summary_type_out.bottom_5_values(4) || '-' ||
                                               summary_type_out.bottom_5_values(5));
END;
/

Summary statistics for column BACKERS in the table DSI_KICKSTARTERSCRAPE_DS_I
Count:        45815
Min:              0
Max:          87142
Range:        87142
Mean:            70
Mode Count:       1
Mode:             0
Variance:    475665
Stddev:         690
Quantile 25:      5
Median:          23
Quantile 75:     59
Top 5 values:    87142-68929-61290-36276-24883
Bottom 5 values: 0-0-0-0-0
```

This is a **skewed** distribution where **mean, median** and **mode** are markedly different.

2. Generating a histogram for the column BACKERS (bin-width: 5000)

```
SELECT bkt, COUNT(*)
FROM
   (
      SELECT backers,
             CASE
                WHEN backers BETWEEN     0 AND  4999 THEN '01.     0 -  4999'
                WHEN backers BETWEEN  5000 AND  9999 THEN '02.  5000 -  9999'
                WHEN backers BETWEEN 10000 AND 14999 THEN '03. 10000 - 14999'
                WHEN backers BETWEEN 15000 AND 19999 THEN '04. 15000 - 19999'
                WHEN backers BETWEEN 20000 AND 24999 THEN '05. 20000 - 24999'
                WHEN backers BETWEEN 25000 AND 29999 THEN '06. 25000 - 29999'
                WHEN backers BETWEEN 30000 AND 34999 THEN '07. 30000 - 34999'
                WHEN backers BETWEEN 35000 AND 39999 THEN '08. 35000 - 39999'
                WHEN backers BETWEEN 40000 AND 44999 THEN '09. 40000 - 44999'
                WHEN backers BETWEEN 45000 AND 49999 THEN '10. 45000 - 49999'
                WHEN backers BETWEEN 50000 AND 54999 THEN '11. 50000 - 54999'
                WHEN backers BETWEEN 55000 AND 59999 THEN '12. 55000 - 59999'
                WHEN backers BETWEEN 60000 AND 64999 THEN '13. 60000 - 64999'
                WHEN backers BETWEEN 65000 AND 69999 THEN '14. 65000 - 69999'
                WHEN backers BETWEEN 70000 AND 74999 THEN '15. 70000 - 74999'
                WHEN backers BETWEEN 75000 AND 79999 THEN '16. 75000 - 79999'
                WHEN backers BETWEEN 80000 AND 84999 THEN '17. 80000 - 84999'
                WHEN backers BETWEEN 85000 AND 89999 THEN '18. 85000 - 89999'
             END AS bkt
        FROM dsi_kickstarterscrape_ds_i
    )
GROUP BY bkt
ORDER BY bkt;
```

# of backers	# of projects
01. 0 - 4999	45783
02. 5000 - 9999	18
03. 10000 - 14999	8
05. 20000 - 24999	2
08. 35000 - 39999	1
13. 60000 - 64999	1
14. 65000 - 69999	1
18. 85000 - 89999	1



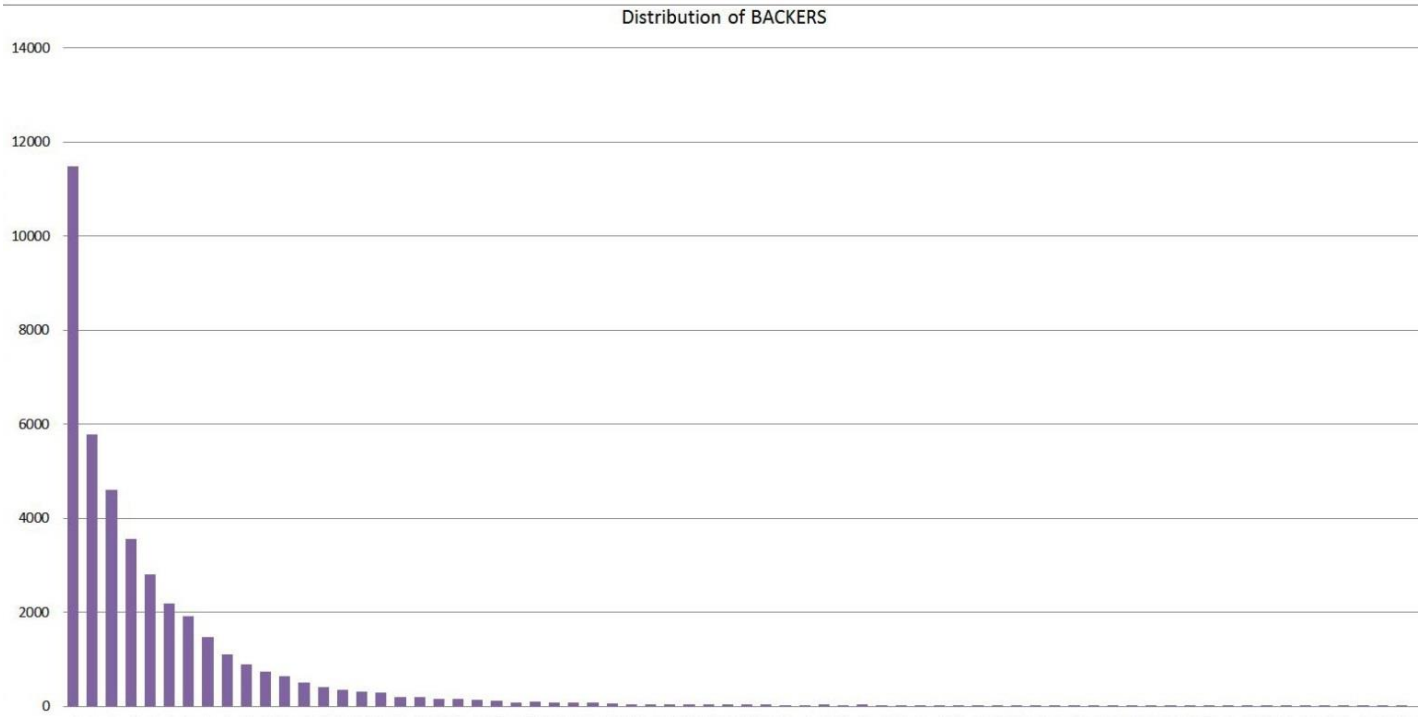
3. Checking outliers.

Checking campaigns with the highest numbers of backers (values that significantly differ from the majority of column values, such as campaign “Double Fine Adventure” with 87,142 backers) shows that in this case, outliers indicate a **heavy-tailed distribution (a** distribution that has **high skewness**) rather than a data error.

This is a **heavily skewed** distribution. It is zero-heavy (3933 campaigns have zero backers**)** and is concentrated around lower values. There are also a few extremely high values. After the column values reach 700, it is sparsely populated (with values ranging out to over 87,000).

```
SELECT buckets, COUNT(*) as row_count
  FROM (SELECT CASE
                 WHEN backers BETWEEN    1 AND    9 THEN  '001.    1 -    9'
                 WHEN backers BETWEEN   10 AND   19 THEN  '002.   10 -   19'
                 WHEN backers BETWEEN   20 AND   29 THEN  '003.   20 -   29'
                 WHEN backers BETWEEN   30 AND   39 THEN  '004.   30 -   39'
                 WHEN backers BETWEEN   40 AND   49 THEN  '005.   40 -   49'
                 WHEN backers BETWEEN   50 AND   59 THEN  '006.   50 -   59'
                 WHEN backers BETWEEN   60 AND   69 THEN  '007.   60 -   69'
                 WHEN backers BETWEEN   70 AND   79 THEN  '008.   70 -   79'
                 WHEN backers BETWEEN   80 AND   89 THEN  '009.   80 -   89'
                 WHEN backers BETWEEN   90 AND   99 THEN  '010.   90 -   99'
                 …
                 WHEN backers BETWEEN  670 AND  679 THEN  '068.  670 -  679'
                 WHEN backers BETWEEN  680 AND  689 THEN  '069.  680 -  689'
                 WHEN backers BETWEEN  690 AND  699 THEN  '070.  690 -  699'
             END AS buckets
          FROM dsi_kickstarterscrape_ds_i)
 GROUP BY buckets
 ORDER BY buckets;
```

If I select only projects with relatively low number of backers (up to 700) and filter out the projects that do not have any backers, the distribution of BACKERS can be represented by the following graph:



4. Calculating skewness

I used the following method (described in Oracle® Crystal Ball Reference and Examples Guide):

Skewness is computed by **finding the third moment about the mean and dividing by the cube of the standard deviation.**

Formula:

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3}$$

```
WITH calc_mean AS
    (SELECT AVG(backers) AS mean
     FROM dsi_kickstarterscrape_ds_i)
SELECT (SUM(POWER((ds.backers - mn.mean), 3)) /
            COUNT(ds.backers)) / POWER(STDDEV(backers), 3) AS skewness
  FROM dsi_kickstarterscrape_ds_i ds, calc_mean mn;

Skewness: 87.34
```

Another way of calculating skewness:

$$Skew = \frac{n}{(n-1)*(n-2)} * \sum_{i=1}^{n}\left(\frac{v_i - \mu}{\sigma}\right)^3$$

```
WITH calc_mean AS
     (SELECT AVG(backers) AS mean,
             STDDEV(backers) AS sd
        FROM dsi_kickstarterscrape_ds_i)
SELECT (COUNT(ds.backers) / ((COUNT(ds.backers) - 1) * (COUNT(ds.backers) - 2))) *
             SUM(POWER(((ds.backers - mn.mean) / mn.sd), 3)) AS skewness
  FROM dsi_kickstarterscrape_ds_i ds,  calc_mean mn;

Skewness: 87.34
```

This value indicates a **highly skewed** distribution.

4. Is the ‘duration’ variable normally distributed?

Answer: No, this is not a normal distribution.

---

1. Collecting statistical data for the column DURATION, using procedure **SUMMARY** from the Oracle built-in package **DBMS_STAT_FUNCS**.

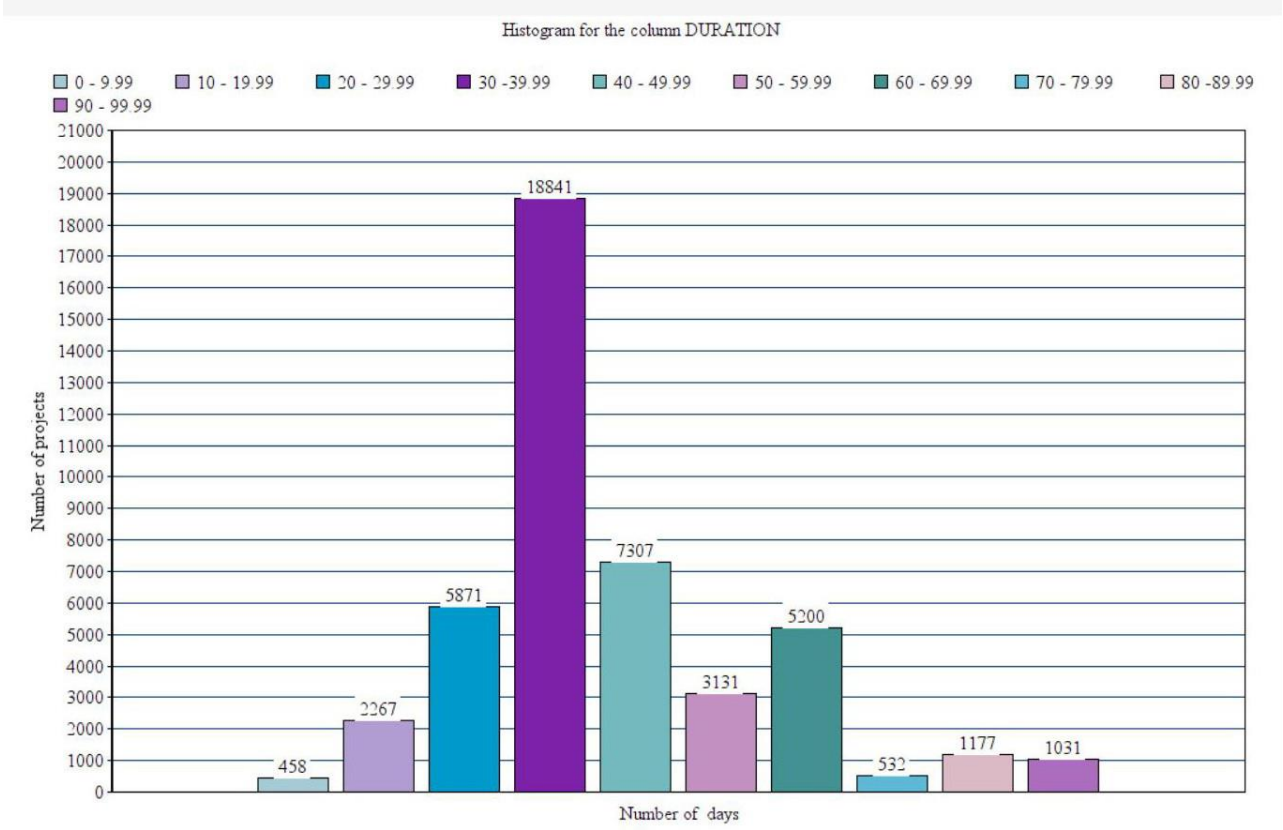
```
DECLARE
   summary_type_out dbms_stat_funcs.SummaryType;
BEGIN
   dbms_stat_funcs.summary('C##DEV', 'DSI_KICKSTARTERSCRAPE_DS_I', 'DURATION', 3,
                                                                    summary_type_out);
   dbms_output.put_line('Summary statistics for column DURATION in the table ' ||
                                                          'DSI_KICKSTARTERSCRAPE_DS_I');
   dbms_output.put_line('Count: '        || summary_type_out.count);
   dbms_output.put_line('Min: '          || summary_type_out.min);
   dbms_output.put_line('Max: '          || summary_type_out.max);
   dbms_output.put_line('Range: '        || summary_type_out.range);
   dbms_output.put_line('Mean: '         || round(summary_type_out.mean));
   dbms_output.put_line('Mode Count: '   || summary_type_out.cmode.count);
   dbms_output.put_line('Mode: '         || summary_type_out.cmode(1));
   dbms_output.put_line('Variance: '     || round(summary_type_out.variance));
   dbms_output.put_line('Stddev: '       || round(summary_type_out.stddev));
   dbms_output.put_line('Quantile 25: '  || summary_type_out.quantile_25);
   dbms_output.put_line('Median: '       || summary_type_out.median);
   dbms_output.put_line('Quantile 75: '  || summary_type_out.quantile_75);
   dbms_output.put_line('Top 5 values: ' || summary_type_out.top_5_values(1) || '-' ||
                                            summary_type_out.top_5_values(2) || '-' ||
                                            summary_type_out.top_5_values(3) || '-' ||
                                            summary_type_out.top_5_values(4) || '-' ||
                                            summary_type_out.top_5_values(5));
   dbms_output.put_line('Bottom 5 values: ' || summary_type_out.bottom_5_values(5) || '-' ||
                                               summary_type_out.bottom_5_values(4) || '-' ||
                                               summary_type_out.bottom_5_values(3) || '-' ||
                                               summary_type_out.bottom_5_values(4) || '-' ||
                                               summary_type_out.bottom_5_values(5));
END;
/

Summary statistics for column DURATION in the table DSI_KICKSTARTERSCRAPE_DS_I
Count:        45815
Min:              1
Max:              91.96
Range:            90.96
Mean:             40
Mode Count:       1
Mode:             30
Variance:         303
Stddev:           17
Quantile 25:      30
Median:           32
Quantile 75:      48.395
Top 5 values:     91.96-91.96-91.96-91.96-91.96
Bottom 5 values: 1-1-1-1-1
```

This is a **skewed** distribution where **mean, median** and **mode** are different.

2. Generating a histogram for the column DURATION (bin-width: 10)

```
SELECT bkt, count(*)
FROM
   (
    SELECT duration,
           CASE
             WHEN duration > 0 AND duration < 10 THEN '0 - 9.99'
             WHEN duration >= 10 AND duration < 20 THEN '10 - 19.99'
             WHEN duration >= 20 AND duration < 30 THEN '20 - 29.99'
             WHEN duration >= 30 AND duration < 40 THEN '30 - 39.99'
             WHEN duration >= 40 AND duration < 50 THEN '40 - 49.99'
             WHEN duration >= 50 AND duration < 60 THEN '50 - 59.99'
             WHEN duration >= 60 AND duration < 70 THEN '60 - 69.99'
             WHEN duration >= 70 AND duration < 80 THEN '70 - 79.99'
             WHEN duration >= 80 AND duration < 90 THEN '80 - 89.99'
             WHEN duration >= 90 AND duration < 100 THEN '90 - 99.99'
             WHEN duration >= 100 THEN 'Over 100'
           END AS bkt
     FROM dsi_kickstarterscrape_ds_i
    )
GROUP BY bkt
ORDER BY bkt;
```

# of days	# of projects
0 - 9.99	458
10 - 19.99	2267
20 - 29.99	5871
30 - 39.99	18841
40 - 49.99	7307
50 - 59.99	3131
60 - 69.99	5200
70 - 79.99	532
80 - 89.99	1177
90 - 99.99	1031



3. Calculating skewness

Method 1.

Formula:

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{s^3}$$

```
WITH calc_mean AS
     (SELECT AVG(duration) AS mean
        FROM dsi_kickstarterscrape_ds_i)
SELECT (SUM(POWER((ds.duration - mn.mean), 3)) / COUNT(ds.duration)) /
          POWER(STDDEV(duration), 3) AS skewness
  FROM dsi_kickstarterscrape_ds_i ds, calc_mean mn;
```

Skewness: 1.08

Method 2.

Formula:

$$Skew = \frac{n}{(n-1)*(n-2)} * \sum_{i=1}^{n}\left(\frac{v_i - \mu}{\sigma}\right)^3$$

```
WITH calc_mean AS
     (SELECT AVG(duration) AS mean,
             STDDEV(duration) AS sd
        FROM dsi_kickstarterscrape_ds_i)
SELECT (COUNT(ds.duration) / ((COUNT(ds.duration) - 1) * (COUNT(ds.duration) - 2))) *
             SUM(POWER(((ds.duration - mn.mean) / mn.sd), 3)) AS skewness
  FROM dsi_kickstarterscrape_ds_i ds,  calc_mean mn;
```

Skewness: 1.08

This value indicates that the distribution is **skewed.**

4. Using Oracle procedure **NORMAL_DIST_FIT** (package **DBMS_STAT_FUNCS**) to test how well values in a column fit a normal distribution.

```
DECLARE
    v_mean      NUMBER;
    v_stdev     NUMBER;
    v_sig       NUMBER;
BEGIN
    SELECT AVG(duration),  STDDEV(duration)
      INTO v_mean, v_stdev
      FROM dsi_kickstarterscrape_ds_i;

    dbms_output.put_line('NORMAL DISTRIBUTION - SHAPIRO_WILKS');

    dbms_stat_funcs.normal_dist_fit(
       'C##DEV', 'DSI_KICKSTARTERSCRAPE_DS_I', 'DURATION', 'SHAPIRO_WILKS', v_mean,
                                                                  v_stdev, v_sig);

    dbms_output.put_line('Mean    : ' || round(v_mean, 4));
    dbms_output.put_line('Stddev  : ' || round(v_stdev, 4));
    dbms_output.put_line('Sig     : ' || to_char(v_sig,'9.9999'));
END;
/

NORMAL DISTRIBUTION - SHAPIRO_WILKS
W value : .88536958097346726539676013231169444482802
Mean    : 39.9951
Stddev  : 17.4206
Sig     :   .0000
```

The Shapiro–Wilk test, which is used for evaluating whether the observations deviate from the normal curve, yields a value equal to **0.885** (*P* < **0.000**), thus, the distribution is **not** normal**.**

5. If you could collect data on another attribute of these projects, what would it be and why?

Answer: I would add a field identifying the creator of the project – to have a possibility to analyse previous projects created by the same creator and to get additional information about the creator. Also, I would like to have information about individual pledges (from a separate dataset, containing project id, backer's id, pledge amount, date and time of each pledge) and about backers (also from a separate dataset) - to get more information about the target audience of each project. I would also like to add fields describing how a project was promoted (platform, number of followers, contacts, number of page views) and a list of keywords describing the project, which might help to understand what attracted backers to a project.

If I could, I would add the following values:

1. A field identifying the **owner (creator) of the project** (preferably a USER_ID). I would like to have a possibility to:
   - retrieve historical data: all projects created by that user – whether they were successful or failed, project goals, number of backers etc
   - join this dataset to another one, containing information about Kickstarter users - that would allow an analyst to retrieve additional information about the project's creator, for example, the date of joining Kickstarter (a more experienced creator can have better chances of success).

2. Information about **individual pledges** – in a separate dataset, providing mapping between PROJECT_ID and USER_ID of each individual backer – project id, backer's id, pledge amount, date and time of each pledge. That might give an analyst an opportunity to find some patterns in project funding. Based on those patterns, some attributes of a project can be adjusted (for example, adding more reward levels).

3. Information about **backers –** in a separate dataset – for example, backer's location (to see if a campaign can reach backers in other states or countries). It would help to get information about each project's target audience.

4. For each project, information about how the project was **promoted** (for example, via social media, e-mail distribution etc)
   - Platform (Twitter, Facebook etc)
   - Number of followers, contacts etc
   - Whether the project was endorsed by Kickstarter
   - For each project page – how much traffic the page received (number of page views etc)

5. For each project, a list of **keywords** or **tags**, collected from the main page, describing what the project is about. These keywords might help to understand what attracted backers to that particular project.

This additional information might help to better predict whether or not a project will be successful in reaching its funding goal.

Part 2: Qualitative Analysis

Create a presentation using Google Slides (max. 5 slides) using the data above (and additional data from those tables) that make clear recommendations on how people can create a successful Kickstarter campaign.

Be sure to consider the following:

- What's the **best length of time** to run a campaign?
- What's the **ideal pledge goal**?
- What **type of projects** would be most successful at getting funded?
- Is there an **ideal month/day/time** to launch a campaign?

Google Slide Link:
`https://docs.google.com/presentation/d/12NSLnZ7oeCEqm96DVyFrMbZadSBDLIe00CfYyh6bEy8/edit#slide=id.p`

1. Preparing data for analysis.

1. Filtering data.

A Kickstarter campaign can have one of the following statuses:

```
SELECT status, COUNT(*) AS num_of_projects
  FROM dsi_kickstarterscrape_ds_i
 GROUP BY status
 ORDER BY 2 DESC;
```

status	num_of_projects
successful	22902
failed	18939
live	3912
canceled	58
suspended	4

I decided to use for analysis only those projects that have reached their deadline and filtered out all projects with the status 'Live'. Also, I filtered out all projects with statuses 'Canceled' and 'Suspended'. The resulting dataset contains only **successful** and **failed** projects.

```
DELETE FROM dsi_kickstarterscrape_ds_i
 WHERE status IN ('live', 'suspended', 'canceled');
```

```
3,974 rows deleted.
```

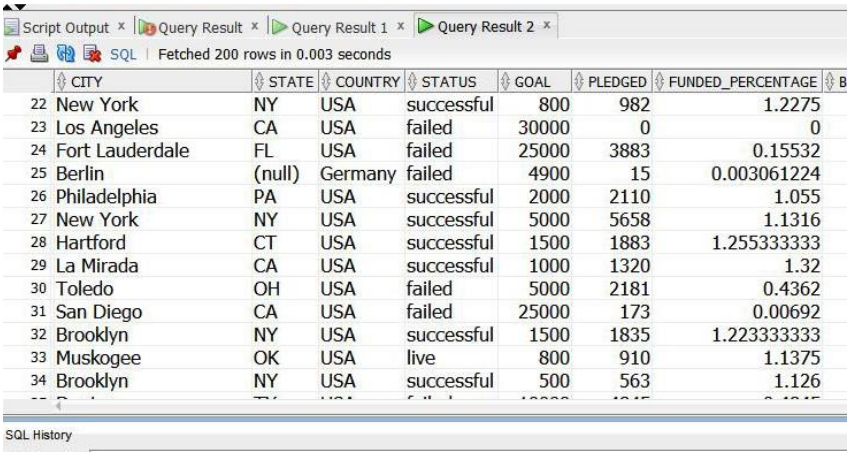
2. Redefining columns.

In the original dataset, LOCATION values are presented as a combination of city and state (or country – for non-USA locations), separated by a comma: for example, **"San Diego, CA"** or **"Buenos Aires, Argentina".** To facilitate data analysis, I replaced LOCATION with 3 separate fields: CITY, STATE and COUNTRY and converted the data type of FUNDED_DATE to TIMESTAMP.

```
CREATE TABLE dsi_kickstarterscrape_ds_sf (
     project_id                  number,
     name                        nvarchar2(500),
     url                        varchar2(200),
     category                    varchar2(100),
     subcategory                 varchar2(100),
     city                        nvarchar2(100),
     state                       char(2),
     country                     nvarchar2(60),
     status                      varchar2(50),
     goal                        number,
     pledged                     number,
     funded_percentage           number,
     backers                     number,
     funded_date                 timestamp,
     levels                     number,
     reward_levels               varchar2(1000),
     updates                     number,
     comments                    number,
     duration                    number
)
/

INSERT INTO dsi_kickstarterscrape_ds_sf
SELECT project_id, name, url, category, subcategory, NULL, NULL, NULL, status, goal, pledged,
       funded_percentage, backers, TO_TIMESTAMP(funded_date, 'Dy, DD Mon YYYY HH24:MI:SS -FF'),
       levels, reward_levels, updates, comments, duration
  FROM dsi_kickstarterscrape_ds_i;
```

```
45,815 rows inserted.
```

```
MERGE INTO dsi_kickstarterscrape_ds_sf ds
USING (SELECT project_id, city,
              CASE
                  WHEN LENGTH(state_or_country) = 2 THEN state_or_country
              END AS state,
              CASE
                  WHEN LENGTH(state_or_country) > 2 THEN state_or_country
              END AS country
         FROM (SELECT project_id,
                      SUBSTR(location, 1, INSTR(location, ',') - 1) AS city,
                      TRIM(SUBSTR(location, INSTR(location, ',') + 1)) AS state_or_country
                 FROM dsi_kickstarterscrape_ds_i)
       ) sq
ON (sq.project_id = ds.project_id)
WHEN MATCHED THEN
   UPDATE
      SET ds.city = sq.city, ds.state = sq.state,
          ds.country = (CASE
                          WHEN sq.state IS NOT NULL THEN TO_NCHAR('USA')
                          ELSE sq.country
                        END);
45,815 rows merged.
```



Script Output | Query Result | Query Result 1 | Query Result 2

SQL | Fetched 200 rows in 0.003 seconds

	CITY	STATE	COUNTRY	STATUS	GOAL	PLEDGED	FUNDED_PERCENTAGE	B
22	New York	NY	USA	successful	800	982	1.2275	
23	Los Angeles	CA	USA	failed	30000	0	0	
24	Fort Lauderdale	FL	USA	failed	25000	3883	0.15532	
25	Berlin	(null)	Germany	failed	4900	15	0.003061224	
26	Philadelphia	PA	USA	successful	2000	2110	1.055	
27	New York	NY	USA	successful	5000	5658	1.1316	
28	Hartford	CT	USA	successful	1500	1883	1.255333333	
29	La Mirada	CA	USA	successful	1000	1320	1.32	
30	Toledo	OH	USA	failed	5000	2181	0.4362	
31	San Diego	CA	USA	failed	25000	173	0.00692	
32	Brooklyn	NY	USA	successful	1500	1835	1.223333333	
33	Muskogee	OK	USA	live	800	910	1.1375	
34	Brooklyn	NY	USA	successful	500	563	1.126	

SQL History

3. Cleaning duplicate values in columns “CATEGORY” and “SUBCATEGORY”.

```
SELECT category, COUNT(*) AS num_projects
  FROM dsi_kickstarterscrape_ds_sf
 GROUP BY category
 ORDER BY 2 DESC;
```

Category	# of projects
Film & Video	12139
Music	10031
Publishing	4150
Art	3684
Theater	2315
Design	1561
Games	1460
Photography	1380
Food	1291
Fashion	1018
Comics	965
Technology	732
Dance	704
Film & Video	411

To correct an obvious data error, I replaced duplicate values '**Film & Video'** and **'Film & Video'** with a single value **'Film and Video':**

```
UPDATE dsi_kickstarterscrape_ds_sf
   SET category = 'Film and Video'
 WHERE category LIKE 'Film%';
```

12,550 rows updated.

Category	# of projects
Film and Video	12550
Music	10031
Publishing	4150
Art	3684
Theater	2315
Design	1561
Games	1460
Photography	1380
Food	1291
Fashion	1018
Comics	965
Technology	732
Dance	704

For the column SUBCATEGORY I replaced the following duplicate values:

1. '**Film & Video'** and **'Film & Video'** with a single value **'Film and Video'**

```
UPDATE dsi_kickstarterscrape_ds_sf
   SET subcategory = 'Film and Video'
 WHERE subcategory LIKE 'Film%';
```

2,333 rows updated.

2. '**Board & Card Games'** and '**Board & Card Games**' with a single value **'Board and Card Games'**

```
UPDATE dsi_kickstarterscrape_ds_sf
        SET subcategory = 'Board and Card Games'
  WHERE subcategory LIKE 'Board%';
```

471 rows updated.

3. 'Country & Folk' and 'Country & Folk' with a single value **'Country and Folk'**

```
UPDATE dsi_kickstarterscrape_ds_sf
     SET subcategory = 'Country and Folk'
 WHERE subcategory LIKE 'Country%';
```

987 rows updated.

2. Exploring attributes of successful and failed projects

```
WITH
  total_rows AS
   (SELECT COUNT(*) AS total_row_couint
    FROM dsi_kickstarterscrape_ds_sf),
  sf_projects AS
   (SELECT INITCAP(status) AS status, COUNT(*) AS num_projects,
          ROUND(SUM(goal)) AS total_goal, ROUND(AVG(goal)) AS avg_goal,
          ROUND(SUM(pledged)) AS total_pledged,  ROUND(AVG(pledged)) AS avg_pledged,
          ROUND(AVG(funded_percentage) * 100) AS avg_funded_pct,
           ROUND(AVG(backers)) AS avg_backers, ROUND(AVG(levels), 1) AS avg_levels,
          ROUND(AVG(updates), 1) AS avg_updates, ROUND(AVG(comments), 1) AS avg_comments,
          ROUND(AVG(duration), 1) AS avg_duration
      FROM dsi_kickstarterscrape_ds_sf
     GROUP BY status)
SELECT sf.status AS source,
       TO_CHAR(sf.num_projects, '999,999') AS num_projects,
       ROUND(sf.num_projects/t.total_row_couint * 100) AS pct,
       TO_CHAR(sf.total_goal, '9,999,999,999') AS total_goal,
       TO_CHAR(sf.avg_goal, '999,999') AS avg_goal,
       TO_CHAR(sf.total_pledged, '9,999,999,999') AS total_pledged,
       TO_CHAR(sf.avg_pledged, '999,999') AS avg_pledged,
       sf.avg_funded_pct AS avg_funded_pct,
       sf.avg_duration AS avg_duration,
       sf.avg_backers AS avg_backers,
       sf.avg_levels as avg_levels,
       sf.avg_updates AS avg_updates,
       sf.avg_comments AS avg_comments
  FROM sf_projects sf, total_rows t
UNION ALL
SELECT 'All' AS source,
       TO_CHAR(COUNT(*), '999,999') AS num_projects,
       100 AS pct,
       TO_CHAR(ROUND(SUM(goal)), '9,999,999,999') AS total_goal,
       TO_CHAR(ROUND(AVG(goal)), '999,999') AS avg_goal,
       TO_CHAR(ROUND(SUM(pledged)), '9,999,999,999') AS total_pledged,
       TO_CHAR(ROUND(AVG(pledged)), '999,999') AS avg_pledged,
       ROUND(AVG(funded_percentage) * 100) AS avg_funded_pct,
       ROUND(AVG(duration), 1) AS avg_duration,
       ROUND(AVG(backers)) AS avg_backers,
       ROUND(AVG(levels), 1) AS avg_levels,
       ROUND(AVG(updates), 1) AS avg_updates,
       ROUND(AVG(comments), 1) AS avg_comments
 FROM dsi_kickstarterscrape_ds_sf;
```

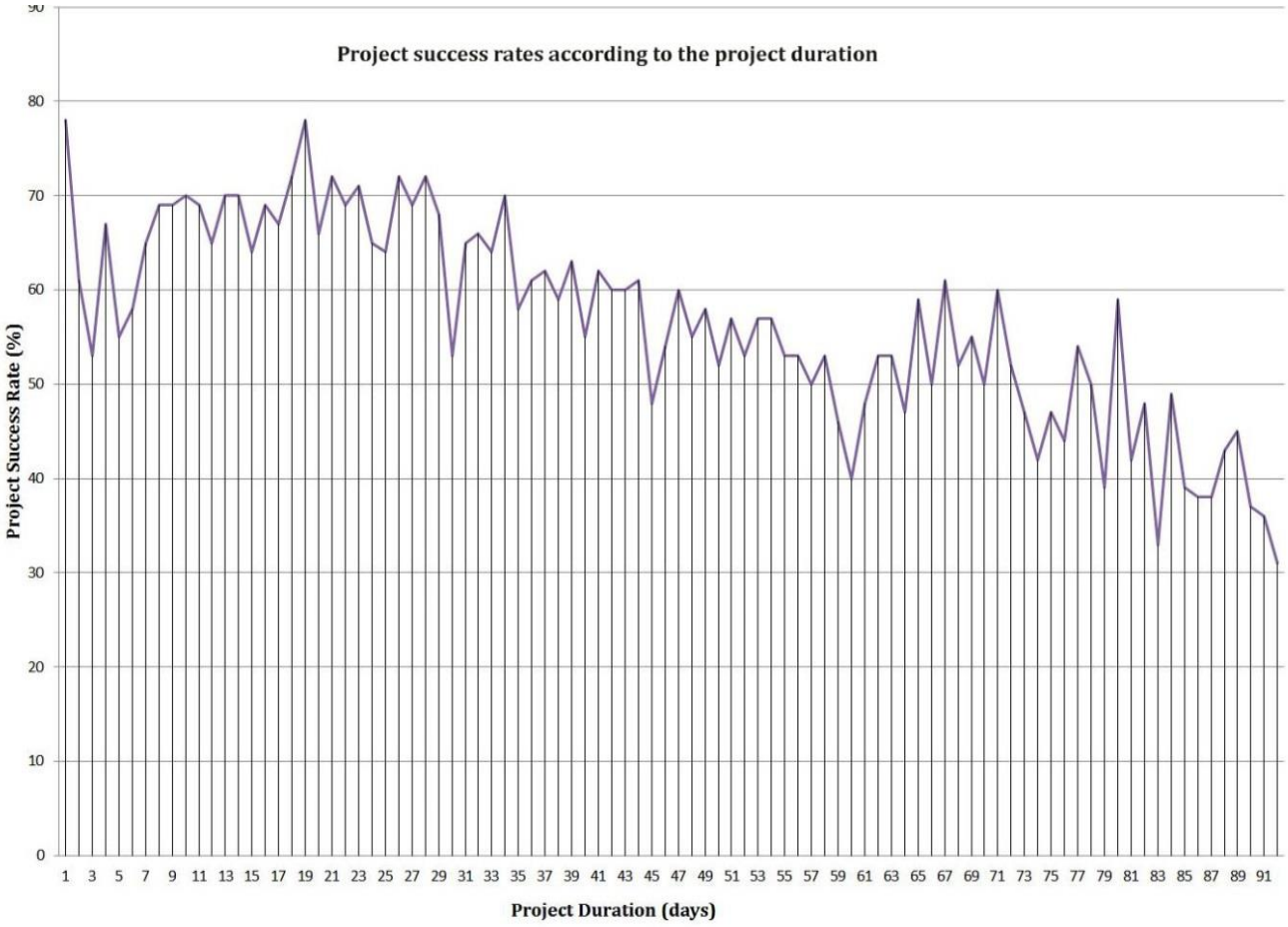
SOURCE	NUM_PROJECTS	PCT	TOTAL_GOAL	AVG_GOAL	TOTAL_PLEDGED	AVG_PLEDGED	AVG_FUNDED_PCT	AVG_DURATION	AVG_BACKERS	AVG_LEVELS	AVG_UPDATES	AVG_COMMENTS
Failed	18,939	45	309,939,807	16,365	16,995,716	897	10	43	13	7.3	1.5	1
Successful	22,902	55	125,659,993	5,487	197,592,764	8,628	353	38	119	8.5	6.7	14.4
All	41,841	100	435,599,800	10,411	214,588,480	5,129	198	40.2	71	8	4.3	8.3

3. What's the best length of time to run a campaign?

Answer: From one week to one month. After one month, the project success rates steadily decline.

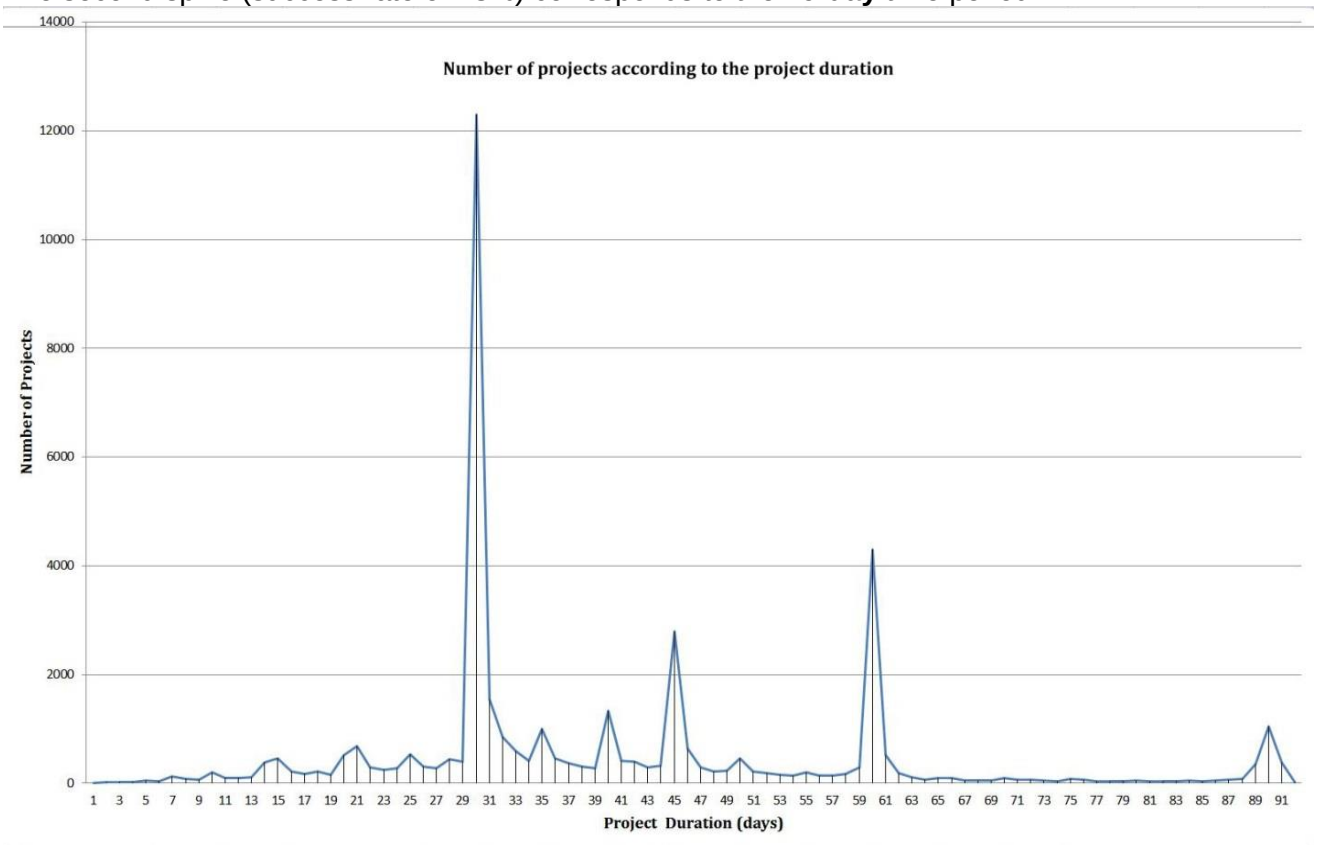
```
SELECT num_days, ROUND((COUNT(success)/COUNT(*)) * 100) AS success_rate,
       COUNT(*) num_projects
  FROM (SELECT ROUND(duration) AS num_days,
               CASE
                  WHEN status = 'successful' THEN 1
               END AS success
          FROM dsi_kickstarterscrape_ds_sf)
 GROUP BY num_days
 ORDER BY num_days;
```

A	B	C	D
Project Duration	Project Success Rate	Number of Projects	
1	78	9	
2	61	18	
3	53	15	
4	67	15	
5	55	44	
6	58	38	
7	65	125	
8	69	81	
9	69	64	
10	70	198	
11	69	93	
12	65	102	
13	70	105	
14	70	385	
15	64	451	
16	69	216	
17	67	176	
18	72	212	
19	78	161	
20	66	518	
21	72	685	
22	69	289	
23	71	244	
24	65	282	
25	64	527	
26	72	307	
27	69	272	
28	72	443	
29	68	390	
30	53	12295	
31	65	1544	



Even though projects running for one day have a success rate of **78%**, very few projects run for one day. The majority of the projects have a duration of **one month**.

The second spike (success rate of **78%**) corresponds to the **19-day** time period.

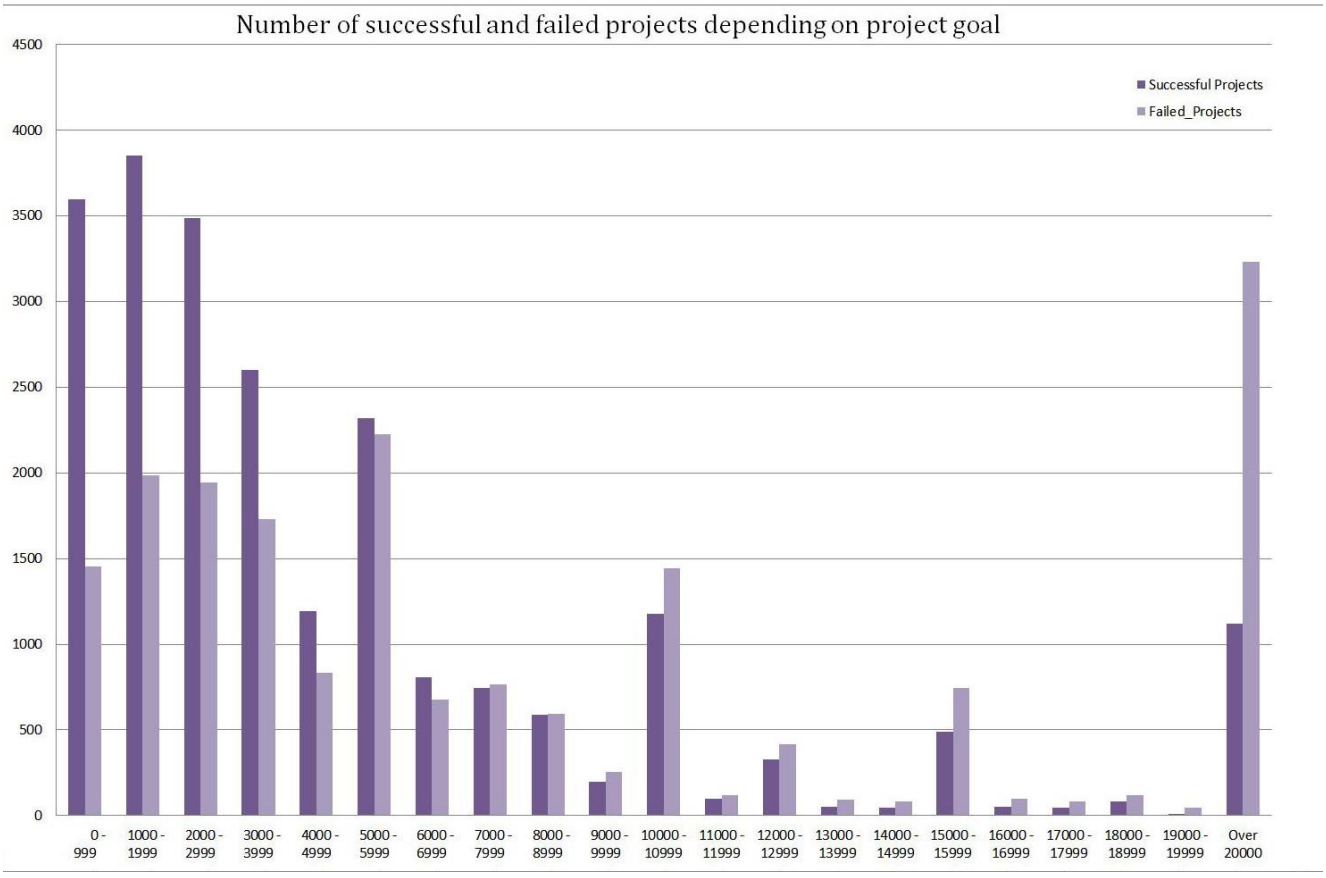


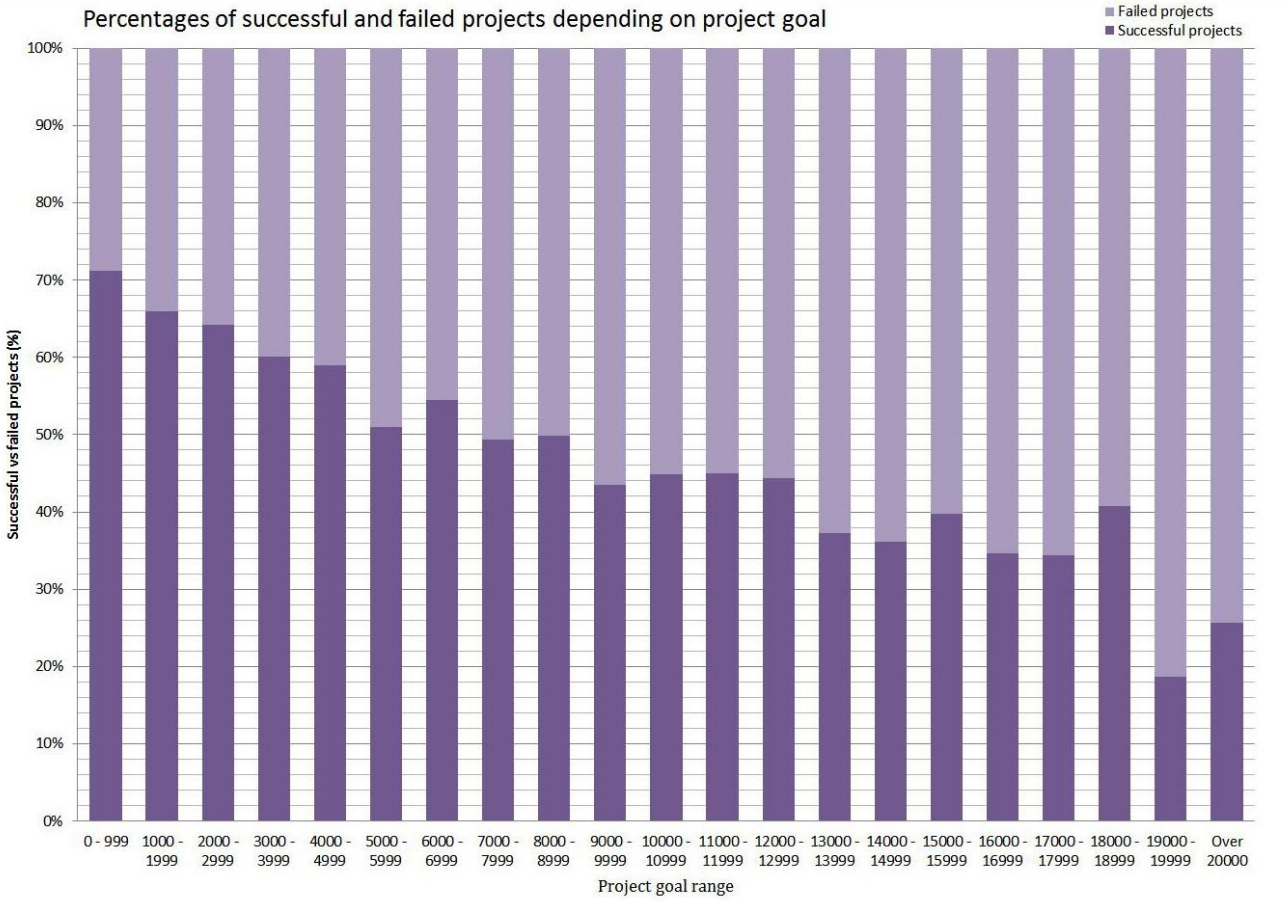
4. What's the ideal pledge goal?

Answer: Ideally, as small as possible. The ideal pledge goal is less than 100 USD – it has the highest success rate – 83%. With a goal amount greater than 7,000 USD, a project has more chances to fail than to succeed.

```
exec generate_query_hist('dsi_kickstarterscrape_ds_sf', 'goal', 20000, 1000)

SELECT SUBSTR(bkt, INSTR(bkt, '.') + 1 ) AS buckets,
    ROUND((COUNT(success)/COUNT(*)) * 100) AS success_rate,
    COUNT(success) AS successful_projects,
    COUNT(failure) AS failed_projects,
    COUNT(*) AS num_projects
 FROM (SELECT CASE
          WHEN goal BETWEEN 0 AND 999 THEN  '001.0 - 999'
          WHEN goal BETWEEN 1000 AND 1999 THEN  '002.1000 - 1999'
          WHEN goal BETWEEN 2000 AND 2999 THEN  '003.2000 - 2999'
          WHEN goal BETWEEN 3000 AND 3999 THEN  '004.3000 - 3999'
          WHEN goal BETWEEN 4000 AND 4999 THEN  '005.4000 - 4999'
          WHEN goal BETWEEN 5000 AND 5999 THEN  '006.5000 - 5999'
          WHEN goal BETWEEN 6000 AND 6999 THEN  '007.6000 - 6999'
          WHEN goal BETWEEN 7000 AND 7999 THEN  '008.7000 - 7999'
          WHEN goal BETWEEN 8000 AND 8999 THEN  '009.8000 - 8999'
          WHEN goal BETWEEN 9000 AND 9999 THEN  '010.9000 - 9999'
          WHEN goal BETWEEN 10000 AND 10999 THEN  '011.10000 - 10999'
          WHEN goal BETWEEN 11000 AND 11999 THEN  '012.11000 - 11999'
          WHEN goal BETWEEN 12000 AND 12999 THEN  '013.12000 - 12999'
          WHEN goal BETWEEN 13000 AND 13999 THEN  '014.13000 - 13999'
          WHEN goal BETWEEN 14000 AND 14999 THEN  '015.14000 - 14999'
          WHEN goal BETWEEN 15000 AND 15999 THEN  '016.15000 - 15999'
          WHEN goal BETWEEN 16000 AND 16999 THEN  '017.16000 - 16999'
          WHEN goal BETWEEN 17000 AND 17999 THEN  '018.17000 - 17999'
          WHEN goal BETWEEN 18000 AND 18999 THEN  '019.18000 - 18999'
          WHEN goal BETWEEN 19000 AND 19999 THEN  '020.19000 - 19999'
          ELSE '99999.Over 20000'
        END AS bkt,
        CASE
          WHEN status = 'successful' THEN 1
        END AS success,
        CASE
          WHEN status = 'failed' THEN 1
        END AS failure
      FROM dsi_kickstarterscrape_ds_sf)
GROUP BY bkt
ORDER BY bkt;
```

BUCKETS	SUCCESS_RATE	SUCCESS...	FAILED...	NUM_PROJ...
0 - 999	71	3599	1453	5052
1000 - 1999	66	3851	1984	5835
2000 - 2999	64	3489	1945	5434
3000 - 3999	60	2603	1729	4332
4000 - 4999	59	1195	832	2027
5000 - 5999	51	2318	2225	4543
6000 - 6999	54	810	678	1488
7000 - 7999	49	746	767	1513
8000 - 8999	50	589	593	1182
9000 - 9999	43	196	255	451
10000 - 10999	45	1177	1443	2620
11000 - 11999	45	100	122	222
12000 - 12999	44	330	414	744
13000 - 13999	37	54	91	145
14000 - 14999	36	48	85	133
15000 - 15999	40	490	744	1234
16000 - 16999	35	52	98	150
17000 - 17999	34	44	84	128
18000 - 18999	41	82	119	201
19000 - 19999	19	11	48	59
Over 20000	26	1118	3230	4348

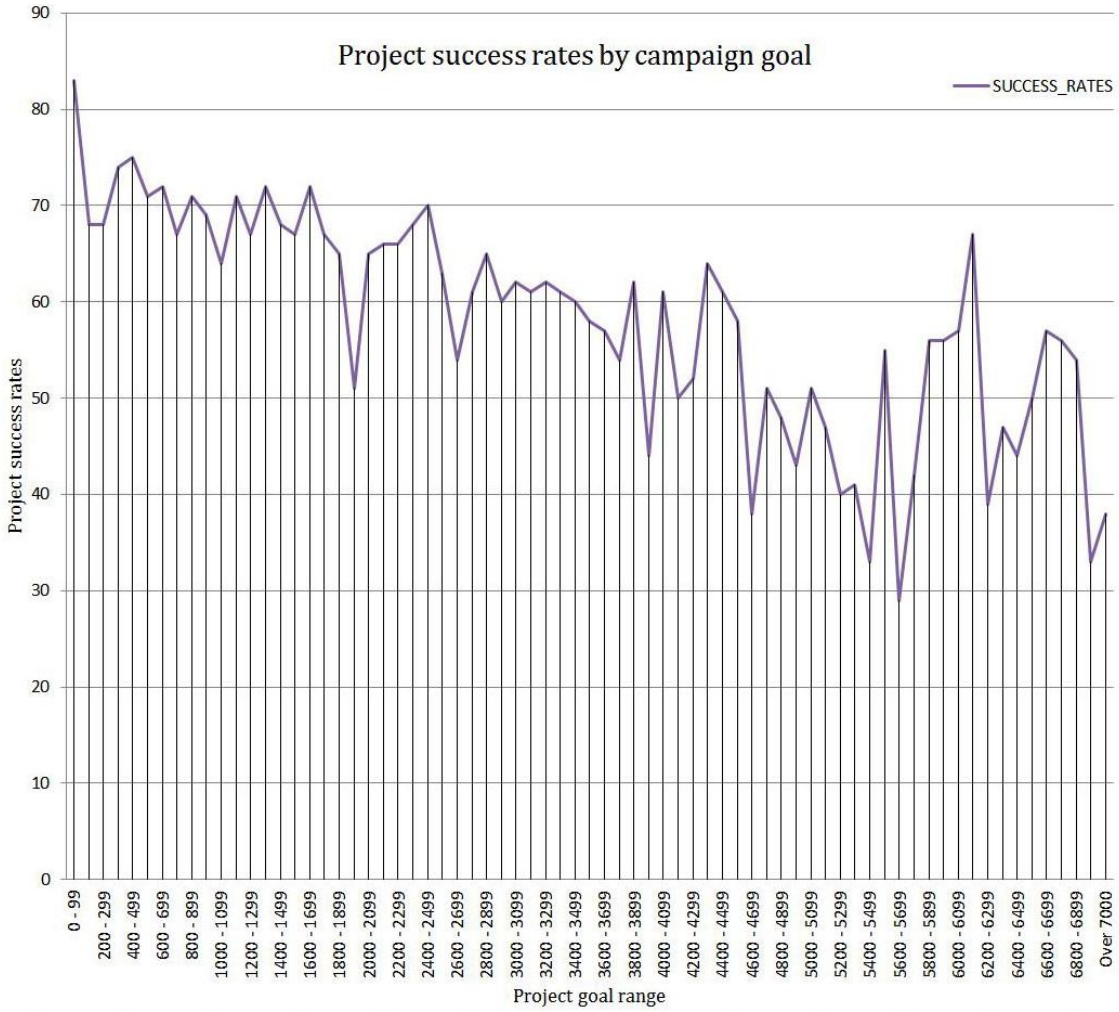




Both graphs show that **the lower the project goal amount is, the higher are its chances to be successful.** With a goal amount greater than **7,000 USD**, a project has more chances to fail than to succeed.

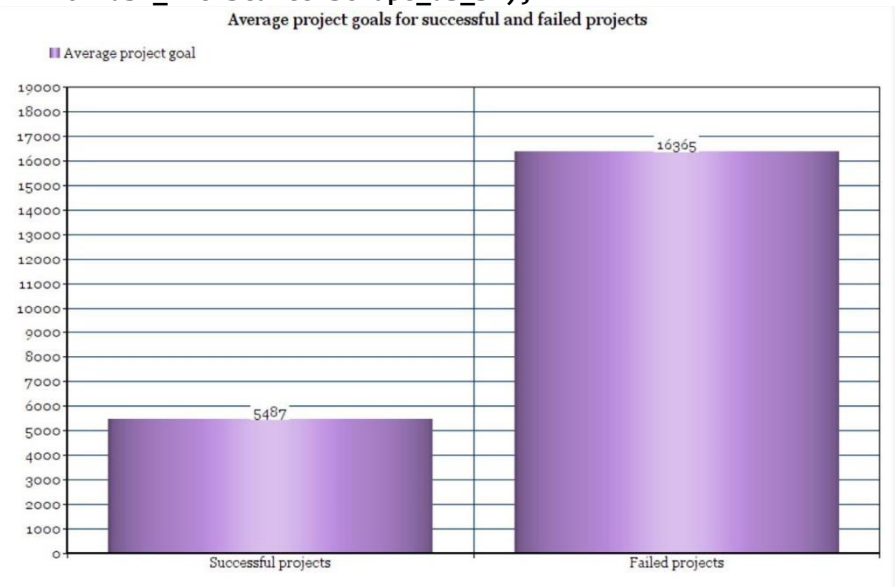
```
exec generate_query_hist('dsi_kickstarterscrape_ds_sf', 'goal', 7000, 100)

SELECT SUBSTR(bkt, INSTR(bkt, '.') + 1 ) AS buckets,
    ROUND((COUNT(success)/COUNT(*)) * 100) AS success_rate,
    COUNT(success) AS successful_projects,
    COUNT(failure) AS failed_projects,
    COUNT(*) AS num_projects
 FROM (SELECT CASE
            WHEN goal BETWEEN 0 AND 99 THEN  '001.0 - 99'
            WHEN goal BETWEEN 100 AND 199 THEN  '002.100 - 199'
            WHEN goal BETWEEN 200 AND 299 THEN  '003.200 - 299'
            WHEN goal BETWEEN 300 AND 399 THEN  '004.300 - 399'
            WHEN goal BETWEEN 400 AND 499 THEN  '005.400 - 499'
            WHEN goal BETWEEN 500 AND 599 THEN  '006.500 - 599'
            WHEN goal BETWEEN 600 AND 699 THEN  '007.600 - 699'
            WHEN goal BETWEEN 700 AND 799 THEN  '008.700 - 799'
            WHEN goal BETWEEN 800 AND 899 THEN  '009.800 - 899'
            WHEN goal BETWEEN 900 AND 999 THEN  '010.900 - 999'
            WHEN goal BETWEEN 1000 AND 1099 THEN  '011.1000 - 1099'
            WHEN goal BETWEEN 1100 AND 1199 THEN  '012.1100 - 1199'
            WHEN goal BETWEEN 1200 AND 1299 THEN  '013.1200 - 1299'
            WHEN goal BETWEEN 1300 AND 1399 THEN  '014.1300 - 1399'
            WHEN goal BETWEEN 1400 AND 1499 THEN  '015.1400 - 1499'
            WHEN goal BETWEEN 1500 AND 1599 THEN  '016.1500 - 1599'
            WHEN goal BETWEEN 1600 AND 1699 THEN  '017.1600 - 1699'
            WHEN goal BETWEEN 1700 AND 1799 THEN  '018.1700 - 1799'
            WHEN goal BETWEEN 1800 AND 1899 THEN  '019.1800 - 1899'
            WHEN goal BETWEEN 1900 AND 1999 THEN  '020.1900 - 1999'
            WHEN goal BETWEEN 2000 AND 2099 THEN  '021.2000 - 2099'
            ...
            WHEN goal BETWEEN 6500 AND 6599 THEN  '066.6500 - 6599'
            WHEN goal BETWEEN 6600 AND 6699 THEN  '067.6600 - 6699'
            WHEN goal BETWEEN 6700 AND 6799 THEN  '068.6700 - 6799'
            WHEN goal BETWEEN 6800 AND 6899 THEN  '069.6800 - 6899'
            WHEN goal BETWEEN 6900 AND 6999 THEN  '070.6900 - 6999'
            ELSE '99999.Over 7000'
        END AS bkt,
        CASE
            WHEN status = 'successful' THEN 1
        END AS success,
        CASE
            WHEN status = 'failed' THEN 1
        END AS failure
    FROM dsi_kickstarterscrape_ds_sf)
GROUP BY bkt
order by bkt;
```

BUCKETS	SUCCESS_RATE	SUCCESS...	FAILED_...	NUM_PROJ...
0 - 99	83	144	29	173
100 - 199	68	154	71	225
200 - 299	68	259	122	381
300 - 399	74	377	134	511
400 - 499	75	249	85	334
500 - 599	71	1066	428	1494
600 - 699	72	385	151	536
700 - 799	67	407	197	604
800 - 899	71	373	151	524
900 - 999	69	185	84	269
1000 - 1099	64	1504	834	2338
1100 - 1199	71	161	67	228
1200 - 1299	67	455	223	678
1300 - 1399	72	82	32	114
1400 - 1499	68	76	36	112
1500 - 1599	67	1161	583	1744
1600 - 1699	72	146	56	202
1700 - 1799	67	106	53	159
1800 - 1899	65	124	67	191
1900 - 1999	51	35	33	68
2000 - 2099	65	1587	862	2449
2100 - 2199	66	60	31	91
2200 - 2299	66	180	94	274
2300 - 2399	68	63	29	92
2400 - 2499	70	71	30	101
2500 - 2599	63	1287	743	2030
2600 - 2699	54	40	34	74
2700 - 2799	61	102	66	168
2800 - 2899	65	73	40	113
2900 - 2999	60	24	16	40
3000 - 3099	62	1549	966	2515
3100 - 3199	61	27	17	44
3200 - 3299	62	90	55	145
3300 - 3399	61	84	54	138
3400 - 3499	60	25	17	42
3500 - 3599	58	657	483	1140
...				
6500 - 6599	50	153	154	307
6600 - 6699	57	12	9	21
6700 - 6799	56	10	8	18
6800 - 6899	54	13	11	24
6900 - 6999	33	5	10	15
Over 7000	38	5040	8095	13135



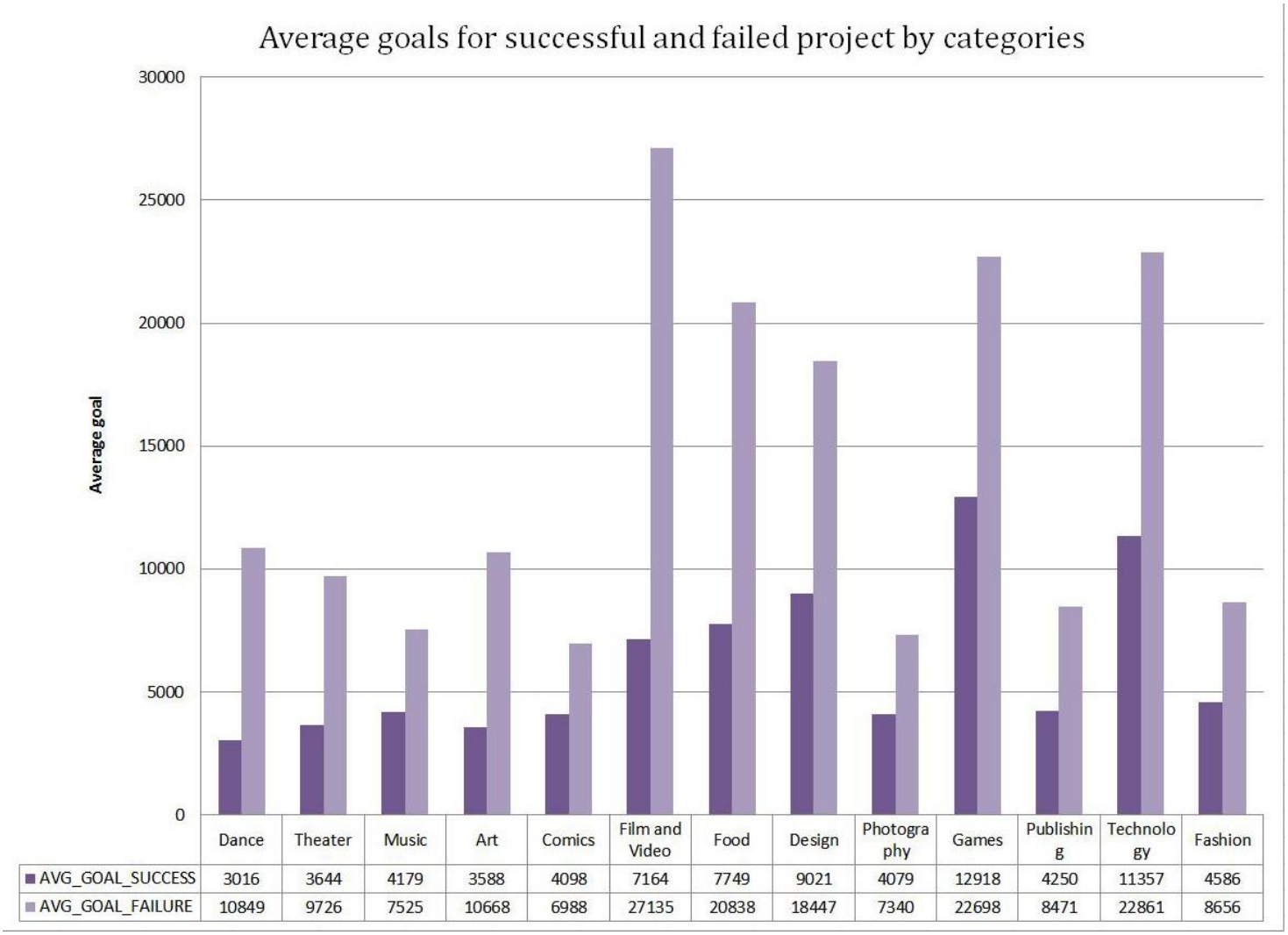
The ideal pledge goal is **less than 100 USD –** it has the **highest success rate – 83%.**

```
SELECT ROUND(AVG(goal_success)) AS avg_goal_success,
       ROUND(AVG(goal_failure)) AS avg_goal_failure
  FROM (SELECT CASE
                  WHEN status = 'successful' THEN ROUND(goal)
               END AS goal_success,
               CASE
                   WHEN status = 'failed' THEN ROUND(goal)
               END AS goal_failure
          FROM dsi_kickstarterscrape_ds_sf);
```



```
SELECT category, ROUND((COUNT(goal_success)/COUNT(*)) * 100) AS success_rate,
       ROUND(AVG(goal_success)) AS avg_goal_success,
       ROUND(AVG(goal_failure)) AS avg_goal_failure
  FROM (SELECT category,
               CASE
                  WHEN status = 'successful' THEN ROUND(goal)
                END AS goal_success,
                CASE
                   WHEN status = 'failed' THEN ROUND(goal)
                END AS goal_failure
          FROM dsi_kickstarterscrape_ds_sf)
 GROUP BY category
 ORDER BY success_rate DESC;
```

	CATEGORY	SUCCESS_RATE	AVG_GOAL_SUCCESS	AVG_GOAL_FAILURE
1	Dance	75	3016	10849
2	Theater	71	3644	9726
3	Music	68	4179	7525
4	Art	57	3588	10668
5	Comics	54	4098	6988
6	Film and Video	51	7164	27135
7	Food	51	7749	20838
8	Design	47	9021	18447
9	Photography	47	4079	7340
10	Games	43	12918	22698
11	Publishing	40	4250	8471
12	Technology	39	11357	22861
13	Fashion	33	4586	8656

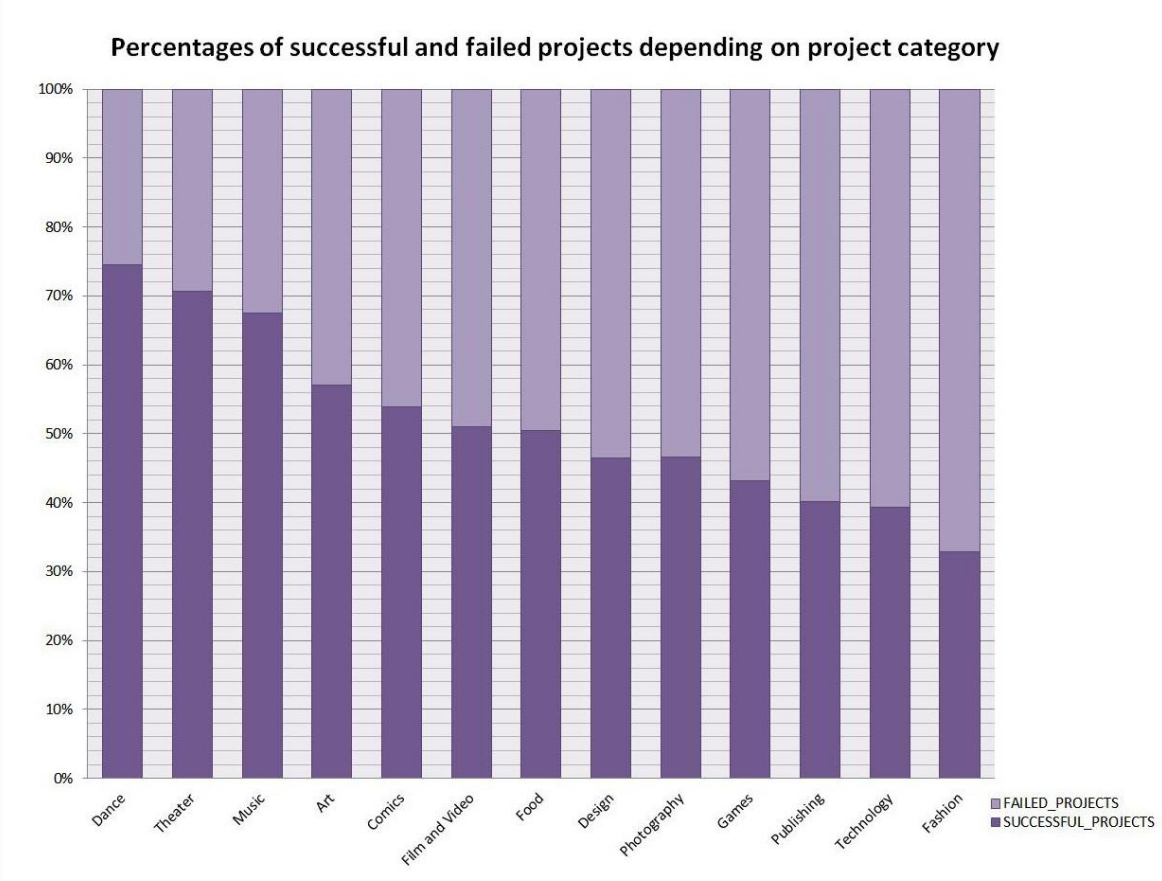


5. What type of projects would be most successful at getting funded?

Answer: Categories DANCE, THEATER and MUSIC would be the most successful at getting funded. If we consider subcategories, then DANCE (not divided into subcategories), MUSIC (INDIE ROCK) and MUSIC (COUNTRY and FOLK) would be the most successful, with success rate 75%.

```
SELECT category, ROUND((COUNT(success)/COUNT(*)) * 100) AS success_rate,
       COUNT(success) AS successful_projects, COUNT(failure) AS failed_projects,
       COUNT(*) num_projects
  FROM (SELECT category,
               CASE
                  WHEN status = 'successful' THEN 1
               END AS success,
               CASE
                  WHEN status = 'failed' THEN 1
               END AS failure
          FROM dsi_kickstarterscrape_ds_sf)
 GROUP BY category
 ORDER BY success_rate desc;
```

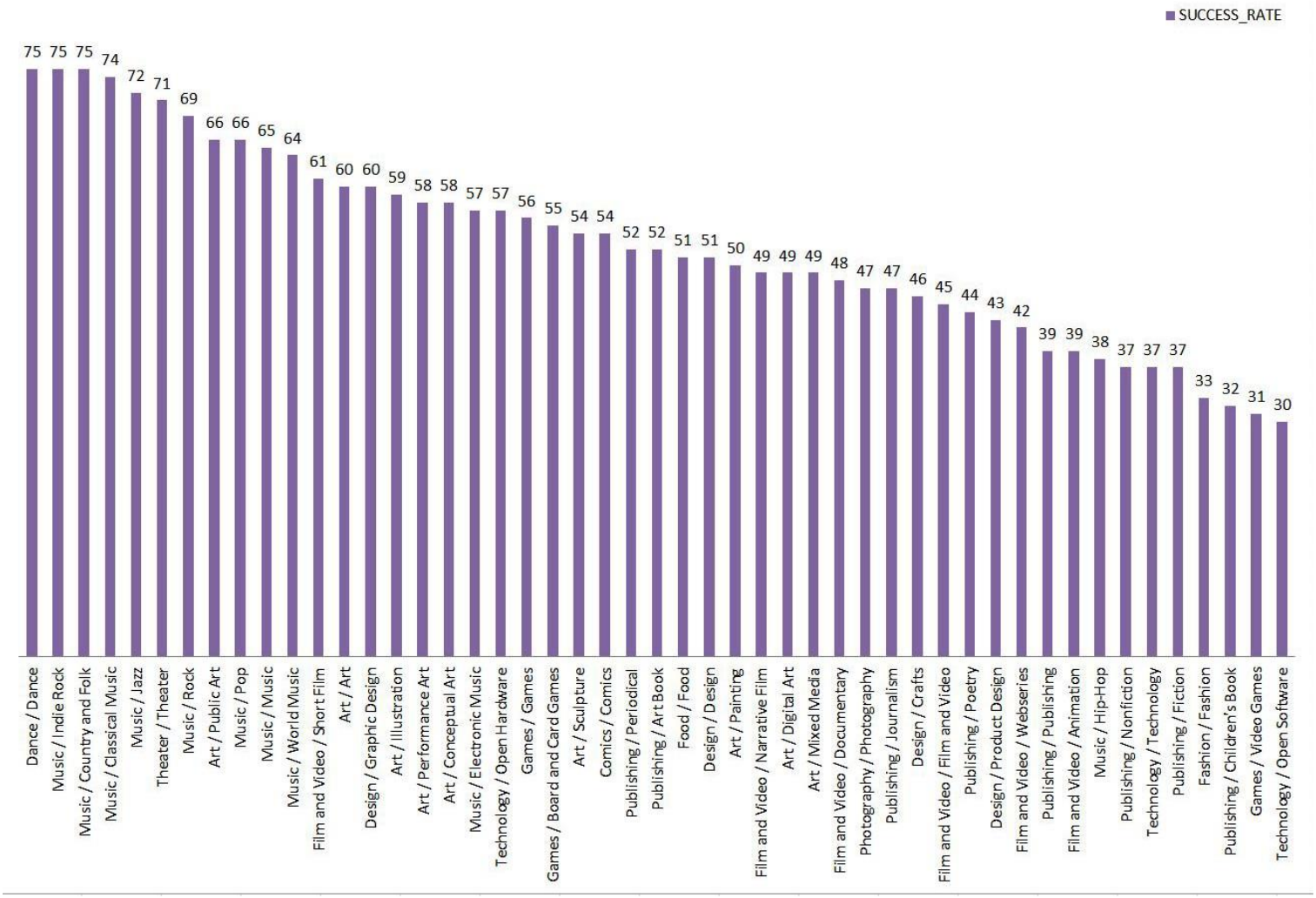
	CATEGORY	SUCCESS_RATE	SUCCESSFUL_PROJECTS	FAILED_PROJECTS	NUM_PROJECTS
1	Dance	75	525	179	704
2	Theater	71	1636	679	2315
3	Music	68	6775	3256	10031
4	Art	57	2102	1582	3684
5	Comics	54	520	445	965
6	Film and Video	51	6400	6150	12550
7	Food	51	652	639	1291
8	Design	47	726	835	1561
9	Photography	47	643	737	1380
10	Games	43	631	829	1460
11	Publishing	40	1669	2481	4150
12	Technology	39	288	444	732
13	Fashion	33	335	683	1018



```
SELECT subcat AS subcategories,
       ROUND((COUNT(success)/COUNT(*)) * 100) AS success_rate,
       COUNT(*) AS num_projects
  FROM (SELECT category || ' / ' || subcategory AS subcat,
               CASE
                  WHEN status = 'successful' THEN 1
              END AS success
          FROM dsi_kickstarterscrape_ds_sf)
 GROUP BY subcat
 ORDER BY success_rate DESC;
```

Category / Subcategory	Success rate	# of projects
Dance / Dance	75	704
Music / Indie Rock	75	1812
Music / Country and Folk	75	987
Music / Classical Music	74	430
Music / Jazz	72	414
Theater / Theater	71	2315
Music / Rock	69	1627
Art / Public Art	66	509
Music / Pop	66	690
Music / Music	65	3000
Music / World Music	64	382
Film and Video / Short Film	61	3735
Art / Art	60	1008
Design / Graphic Design	60	163
Art / Illustration	59	187
Art / Performance Art	58	460
Art / Conceptual Art	58	167
Music / Electronic Music	57	269
Technology / Open Hardware	57	170
Games / Games	56	239
Games / Board and Card Games	55	471
Art / Sculpture	54	329
Comics / Comics	54	965
Publishing / Periodical	52	263
Publishing / Art Book	52	301
Food / Food	51	1291
Design / Design	51	260
Art / Painting	50	483
Film and Video / Narrative Film	49	1408
Art / Digital Art	49	134
Art / Mixed Media	49	407
Film and Video / Documentary	48	3674
Photography / Photography	47	1380
Publishing / Journalism	47	404
Design / Crafts	46	228
Film and Video / Film and Video	45	2333
Publishing / Poetry	44	200
Design / Product Design	43	910
Film and Video / Webseries	42	1024
Publishing / Publishing	39	616
Film and Video / Animation	39	376
Music / Hip-Hop	38	420
Publishing / Nonfiction	37	881
Technology / Technology	37	328
Publishing / Fiction	37	943
Fashion / Fashion	33	1018
Publishing / Children's Book	32	542
Games / Video Games	31	750
Technology / Open Software	30	234

Project success rates by project subcategory



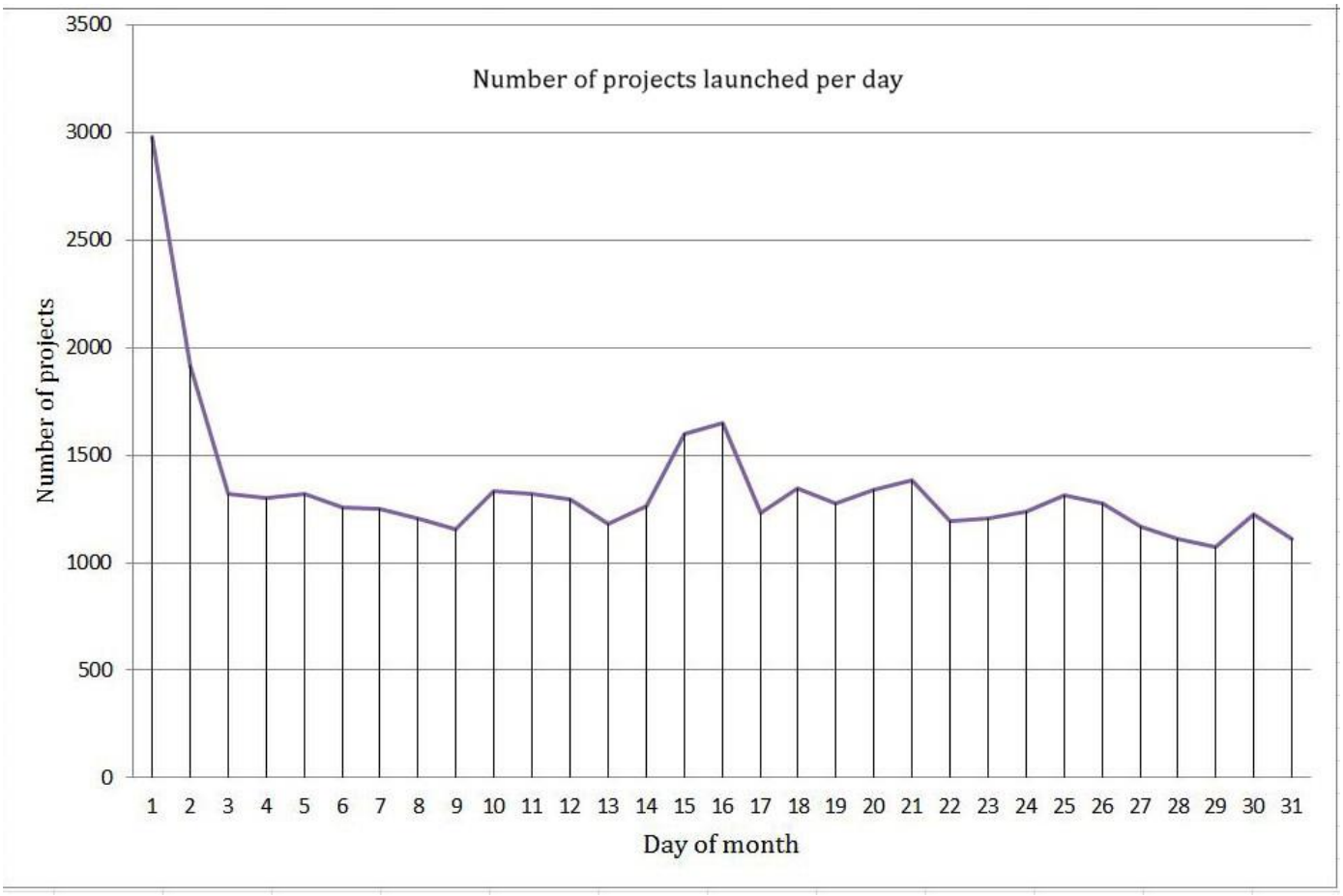
6. Is there an ideal month/day/time to launch a campaign?

Answer: Launching a campaign at the end of a month, and especially at the end of a year, would decrease its chances of successfully reaching its financial goal.

The best months to launch a campaign are MARCH, APRIL and NOVEMBER, and the worst month is DECEMBER. It appears that the best day to start a project is the 12-th day of a month. The days to avoid: 10-th, 20-th, 30-th days of a month as well as the last 5 days of a month.

```
SELECT day_of_month, ROUND((COUNT(success)/COUNT(*)) * 100) AS success_rate,
       COUNT(*) AS num_projects
  FROM (SELECT EXTRACT(DAY FROM funded_date) AS day_of_month,
               CASE
                  WHEN status = 'successful' THEN 1
               END AS success
          FROM dsi_kickstarterscrape_ds_sf)
 GROUP BY day_of_month
 ORDER BY success_rate DESC;
```

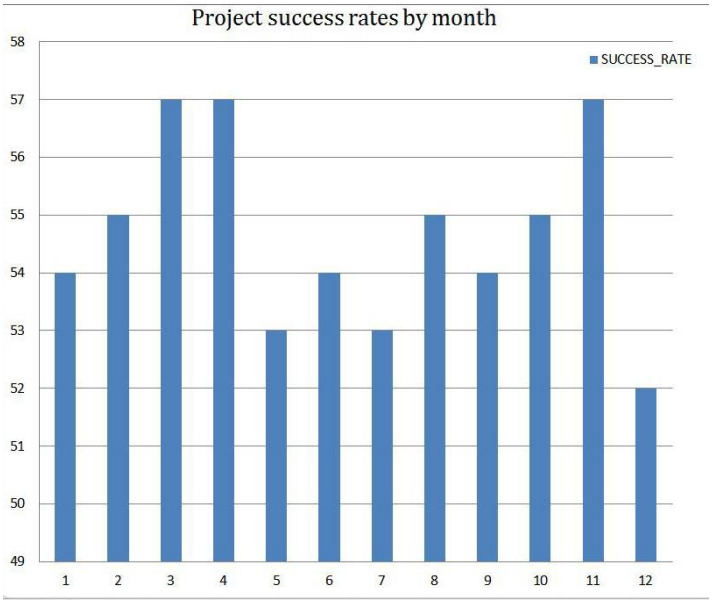
DAY_OF_MONTH	SUCCESS_RATE	NUM_PROJECTS
12	60	1292
5	58	1317
2	58	1916
1	57	2978
16	57	1648
9	57	1157
23	56	1207
17	56	1232
3	56	1321
21	56	1386
15	56	1601
19	55	1273
13	55	1180
14	54	1265
22	54	1192
4	54	1301
25	54	1316
18	54	1344
6	54	1254
8	53	1207
26	53	1278
11	53	1321
24	53	1238
7	53	1252
20	52	1341
10	52	1333
30	52	1223
27	52	1168
29	51	1076
31	51	1114
28	51	1110



It appears that the best day to start a project is the **12-th day** of a month.
The graph also shows that it is better **not** to start a project on the **10-th, 20-th, 30-th days** of a month as well as in **the last 5 days** of a month.

```
SELECT month, ROUND((COUNT(success)/COUNT(*)) * 100) AS success_rate,
        COUNT(*) AS num_projects
   FROM (SELECT EXTRACT(MONTH FROM funded_date) AS month,
                CASE
                   WHEN status = 'successful' THEN 1
                END AS success
           FROM dsi_kickstarterscrape_ds_sf)
  GROUP BY month
  ORDER BY success_rate desc;
```

MONTH	SUCCESS_RATE	NUM_PROJECTS
4	57	5108
3	57	4539
11	57	2634
2	55	3224
10	55	2789
8	55	3037
1	54	3093
6	54	2840
9	54	2748
7	53	2841
5	53	5696
12	52	3292



The best months to launch a campaign are **March, April** and **November**, and the worst month is **December**.