Space missions

from 1957

**data analysis**

22.01.2022

**PROJECT REPORT**

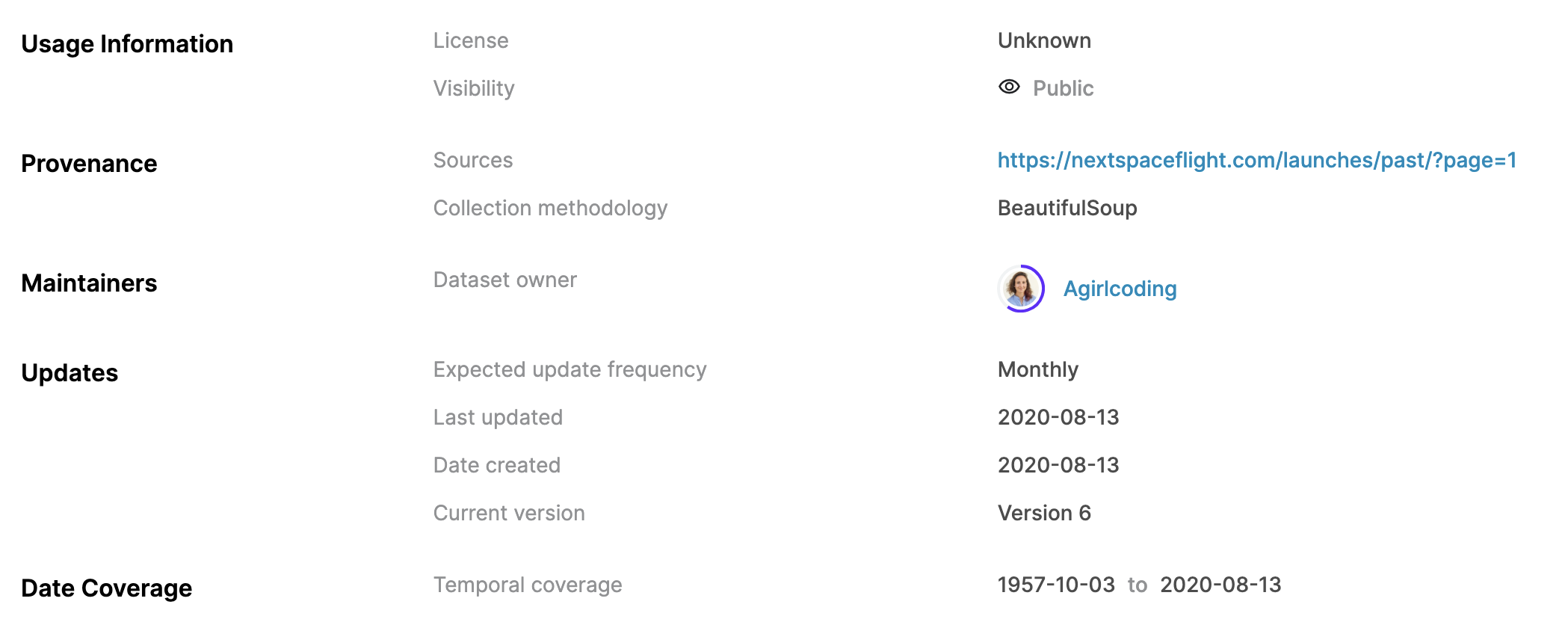


## BUSINESS AREA

In order to make a good and precise analysis on any subject, data analyst must have enough domain knowledge. My main interest is Space exploration and technology. For this reason, the topic of my project is “**Space missions from 1975**”.

## data

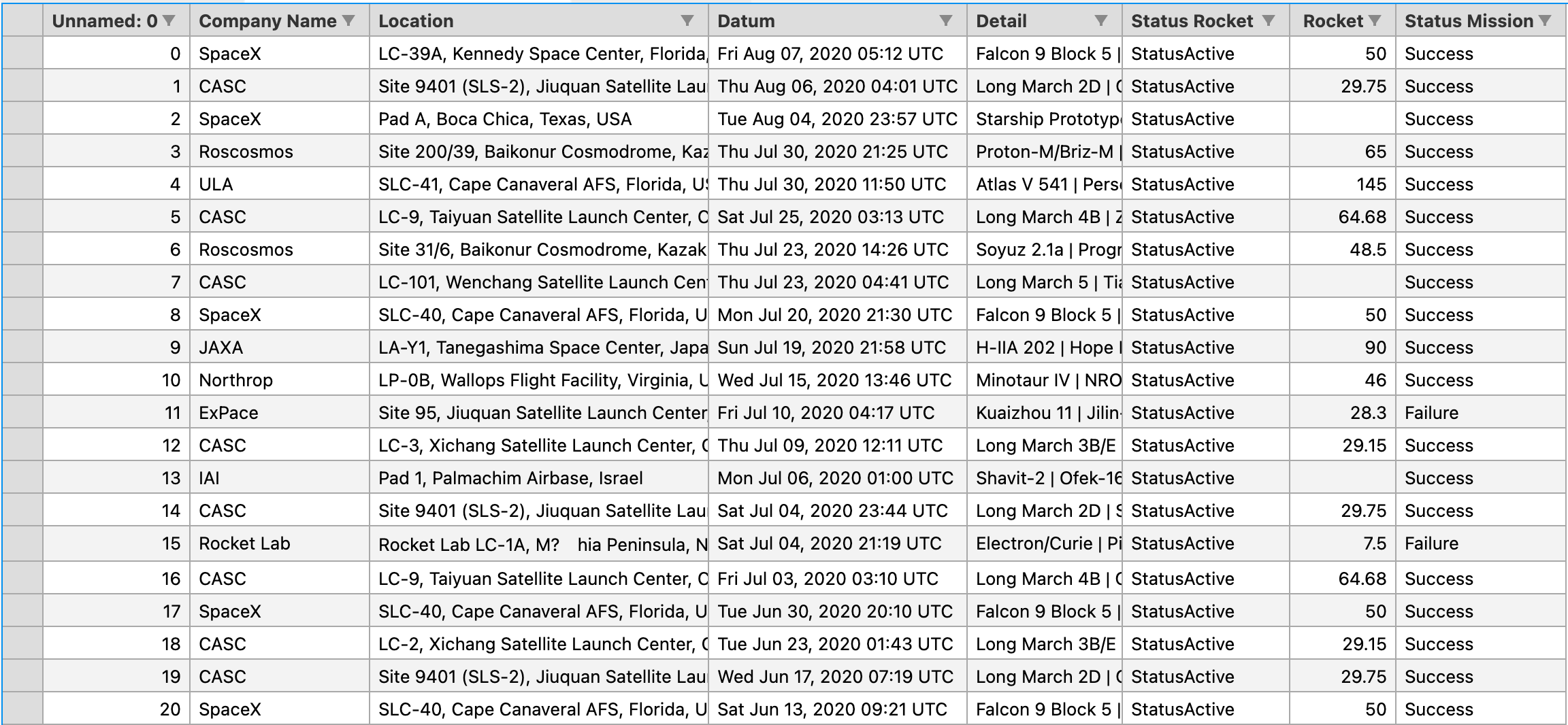




**URL:** <https://www.kaggle.com/agirlcoding/all-space-missions-from-1957>

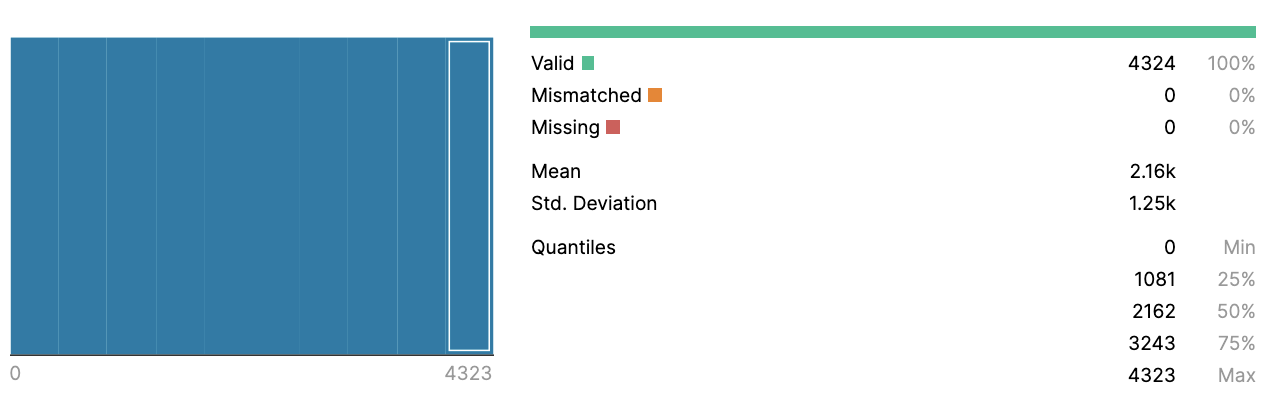
Initial dataset is in .csv (comma separated values) format and of size 103 kB. File consists of 4323 unique records with 8 columns. 5 of them are categorical, 1 of date type and 1 of numerical.

**CSV (preview):**

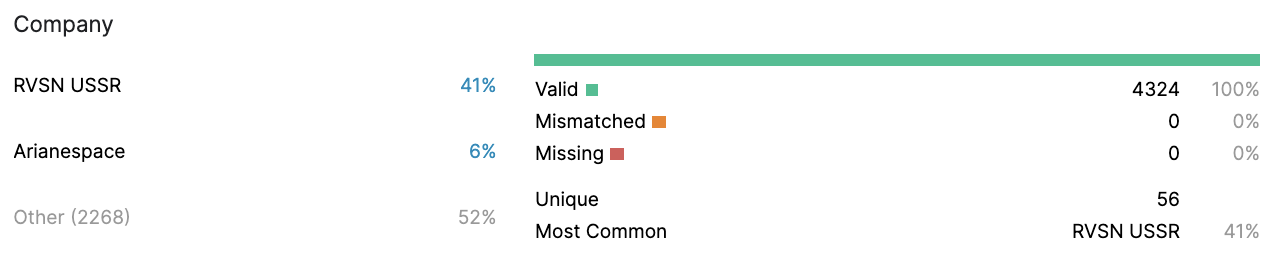


**COLUMNS:**

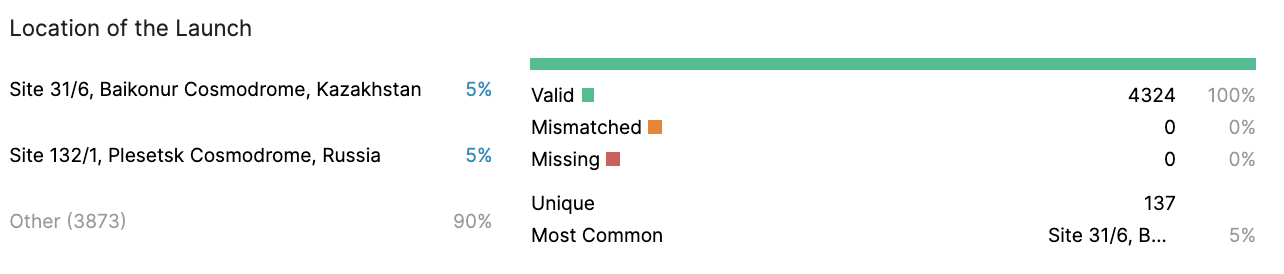
**Unnamed: 0** represents record id within dataset and of type integer:



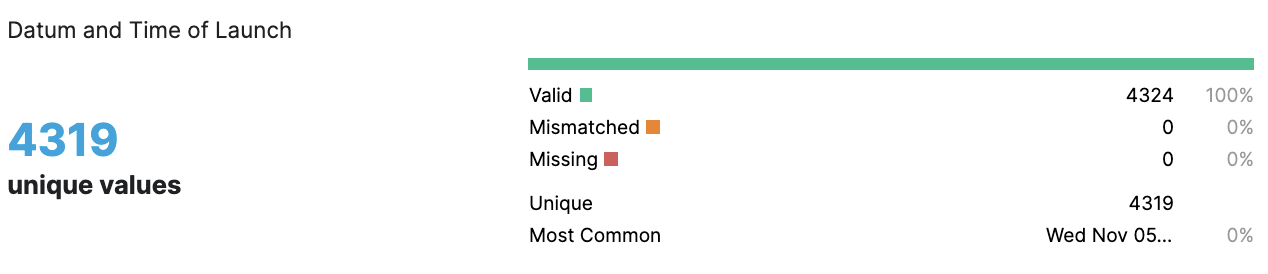
**Company Name** represents names of rocket companies within dataset and of type string:



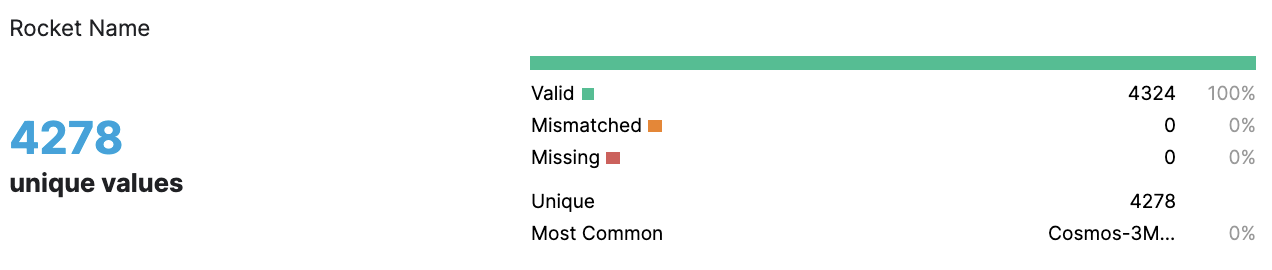
**Location** represents Location of the Launch within dataset and of type string:



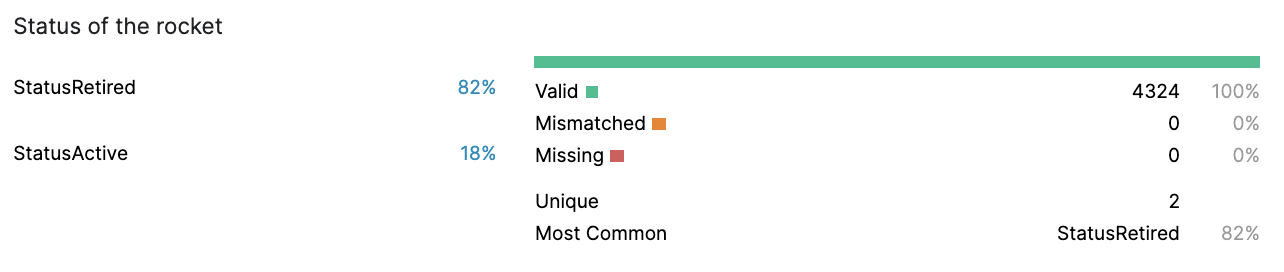
**Datum** represents Date and Time of Launch within dataset and of type datetime:



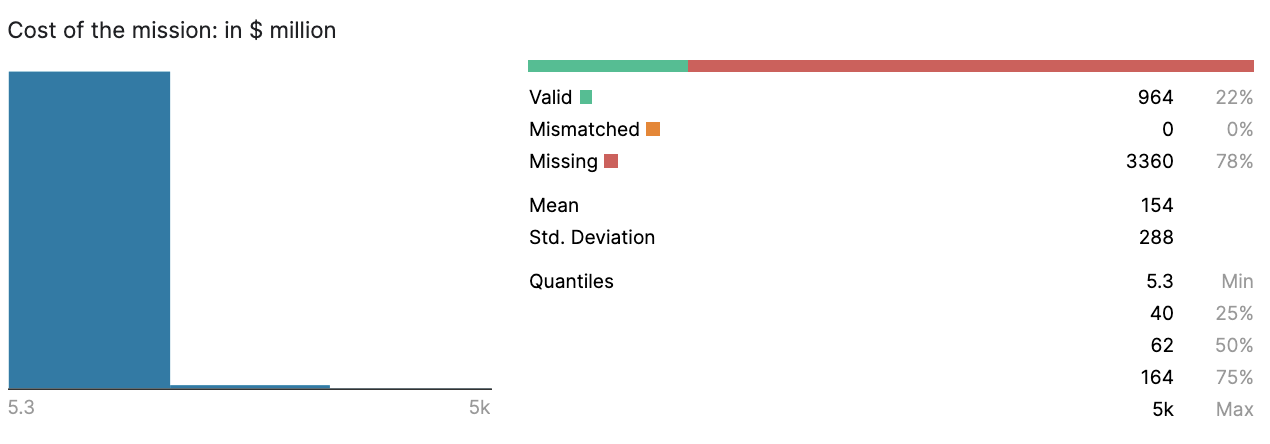
**Detail** represents rocket name and object that was launched within dataset and of type string:



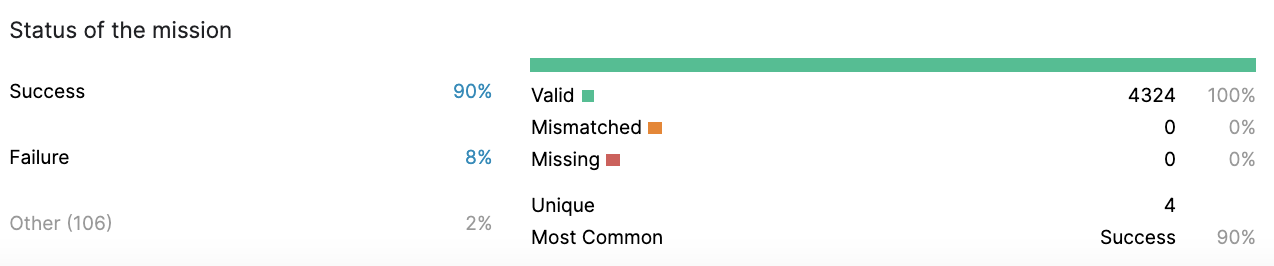
**Status Rocket** represents status of the rocket (active/retired) within dataset and of type string:

****

**Rocket** represents cost of the mission: in $ million within dataset and of type numeric:



**Status Mission** represents status of the mission (success/failure) within dataset and of type string:



## data pre-processing

**Create separate database “space\_missions\_OLTP” and “sp\_missions” schema:**

create database "space\_missions\_OLTP" with owner postgres;

comment on database "space\_missions\_OLTP" is 'OLTP database for space mission analysis';

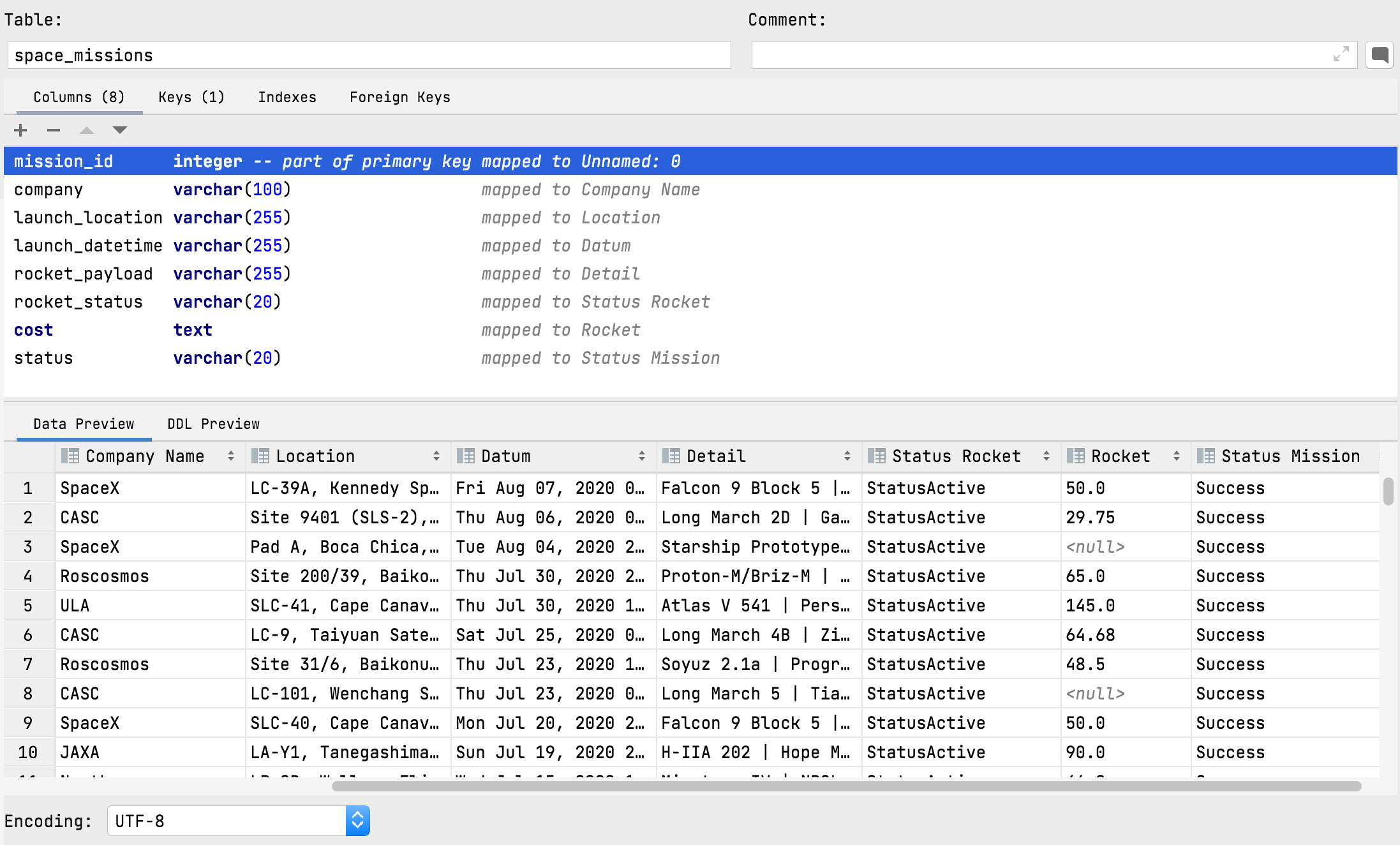
create schema sp\_missions;

alter schema sp\_missions owner to postgres;

set search\_path to sp\_missions;

**Loading data into PostgreSQL:**

For importing .csv data into our database, I have used DataGrip’s data import wizard. The tool automatically creates table for us. I have only specified primary key and column names for resulting table:

****

As a result, we have following table DDL:

create table sp\_missions.space\_missions

(

mission\_id integer not null constraint space\_missions\_pk primary key,

company varchar(100),

launch\_location varchar(255),

launch\_datetime varchar(255),

rocket\_payload varchar(255),

rocket\_status varchar(20),

cost text,

status varchar(20)

);

**DDL:**

Rename table from space\_missions to mission:

alter table space\_missions rename to mission;

Update column launch\_location to contain only main location (country, ocean, sea) of mission:

update mission set launch\_location = TRIM(SPLIT\_PART(launch\_location, ',', -1));

Rename column launch\_location to mission\_location:

alter table mission rename column launch\_location to mission\_location;

Translate launch\_datetime column mission\_date:

launch\_datetime = (Fri Aug 07, 2020 05:12 UTC) and text type

mission\_date = (2020-08-07) and date type:

update mission set launch\_datetime = REPLACE (launch\_datetime, ',', '');

update mission set launch\_datetime = split\_part(launch\_datetime, ' ', 3)

||' '||split\_part(launch\_datetime, ' ', 2)

||' '||split\_part(launch\_datetime, ' ', 4);

alter table mission add column mission\_date date;

update mission set mission\_date = TO\_DATE(launch\_datetime, 'DD Mon YYYY');

alter table mission drop column launch\_datetime;

Column rocket\_payload containt data of format ROCKET | PAYLOAD OBJECT:

We are interested in rockets only so we update column to contain only required data:

alter table mission rename column rocket\_payload to rocket;

update mission set rocket = TRIM(split\_part(rocket, '|', 1));

Our table contains rocket\_status column with 2 distinct values of StatusRetired and StatusActive

We first replace these values to retired and active respectively

We then translate that data into boolean column is\_rocket\_active

Finally we drop rocket\_status column that was replaced by is\_rocket\_active column:

update mission

set rocket\_status = case when rocket\_status = 'StatusRetired' then 'retired'

when rocket\_status = 'StatusActive' then 'active' end;

alter table mission add column is\_rocket\_active boolean;

update mission set is\_rocket\_active = case when rocket\_status = 'retired' then false when rocket\_status = 'active' then true end;

alter table mission drop column rocket\_status;

There some cost values with commas.

In additon it is unreal to make calculations on numbers in text format.

We cannon cast such records into numeric format so we have to eliminate commas in cost column:

update mission set cost = replace(cost, ',', '');

We then create new column for mission cost and populate it with cost values in numeric format:

alter table mission add column mission\_cost numeric;

update mission set mission\_cost = cast(cost as numeric);

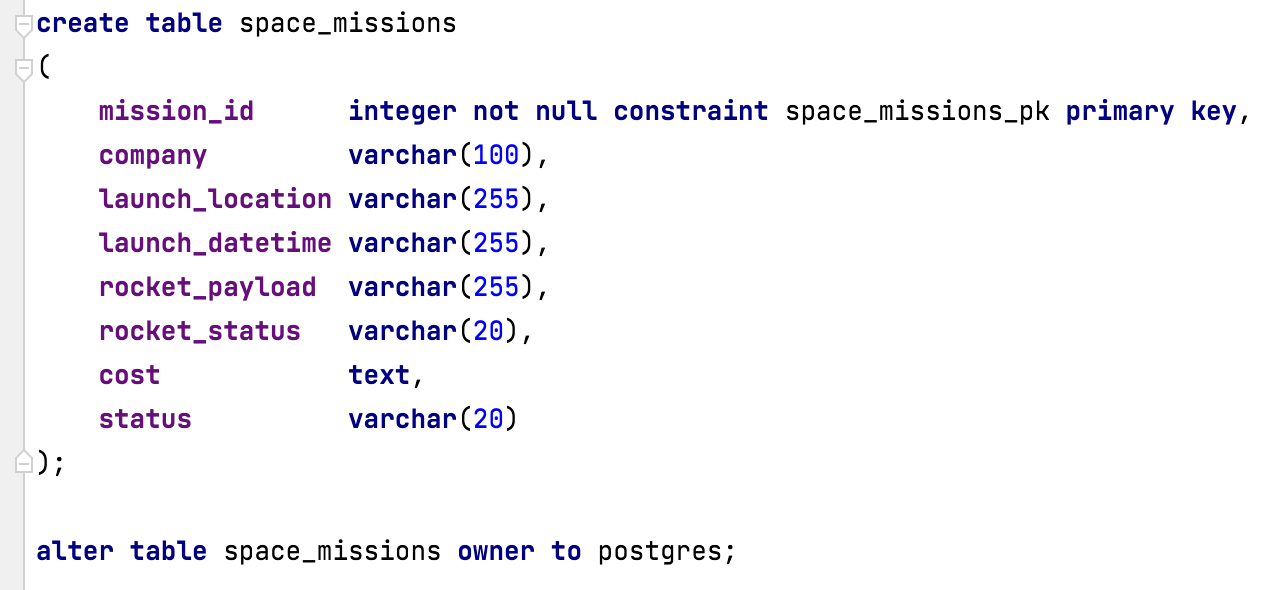
Finally we drop cost column as it was replaced with mission\_cost column:

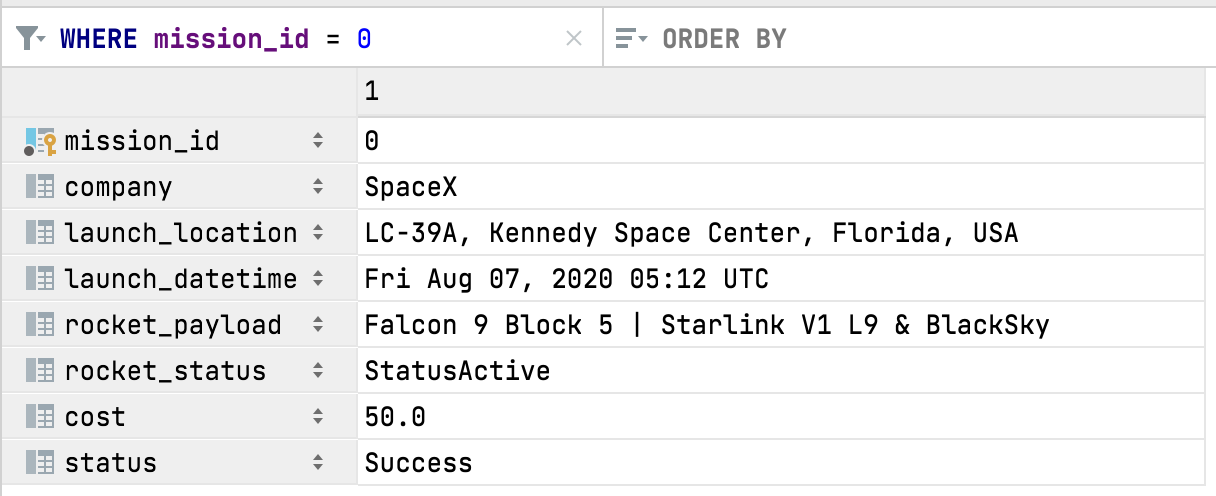
alter table mission drop column cost;

Rename column mission\_cost to cost:

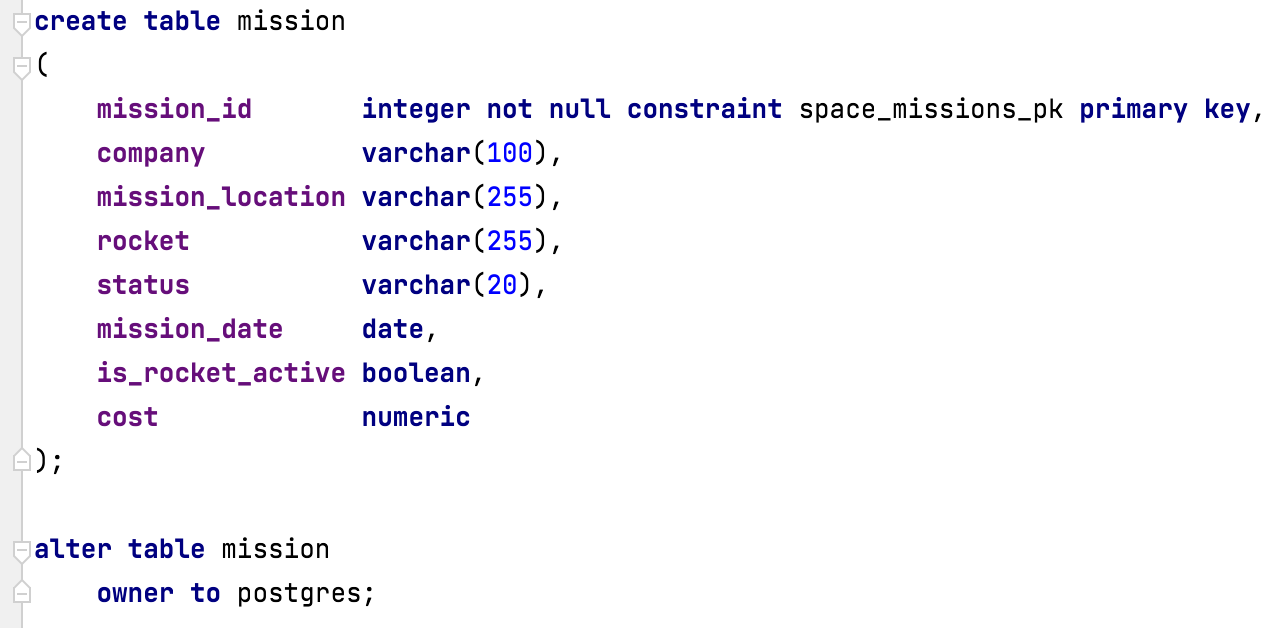
alter table mission rename column mission\_cost to cost;

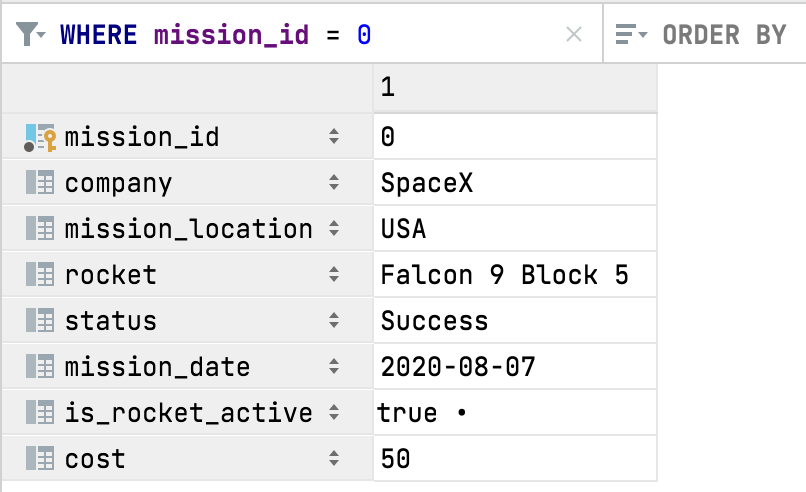
**Before processing:**





**After processing:**





**NULL values:**

Lets check again column for null values:

|  |  |
| --- | --- |
| **Table name** | # NULL records |
| select count(\*) from mission where mission\_id is null; | 0 |
| select count(\*) from mission where company is null; | 0 |
| select count(\*) from mission where mission\_location is null; | 0 |
| select count(\*) from mission where rocket is null; | 0 |
| select count(\*) from mission where status is null; | 0 |
| select count(\*) from mission where mission\_date is null; | 0 |
| select count(\*) from mission where is\_rocket\_active is null; | 0 |
| select count(\*) from mission where cost is null; | 3360 |

There many null values in cost column. There are some techniques to handle such cases such as mean/mode/median imputation, assigning an unique category, search imputation or simply deleting such records.

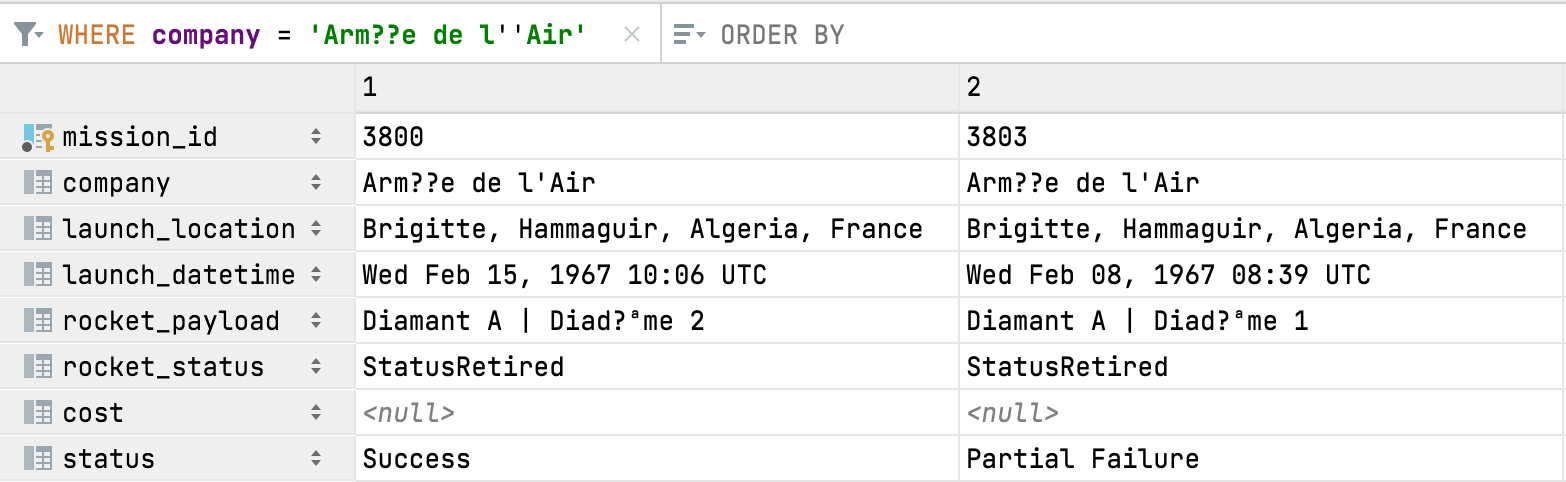
In our case we need precise analysis that is viable. So, we will just take records with nonempty cost column for our analytical queries in later stages.

**Wrong values:**

Check company names for wrong values:

select distinct company from mission;

We have one company with strange value: ("Arm??e de l'Air")



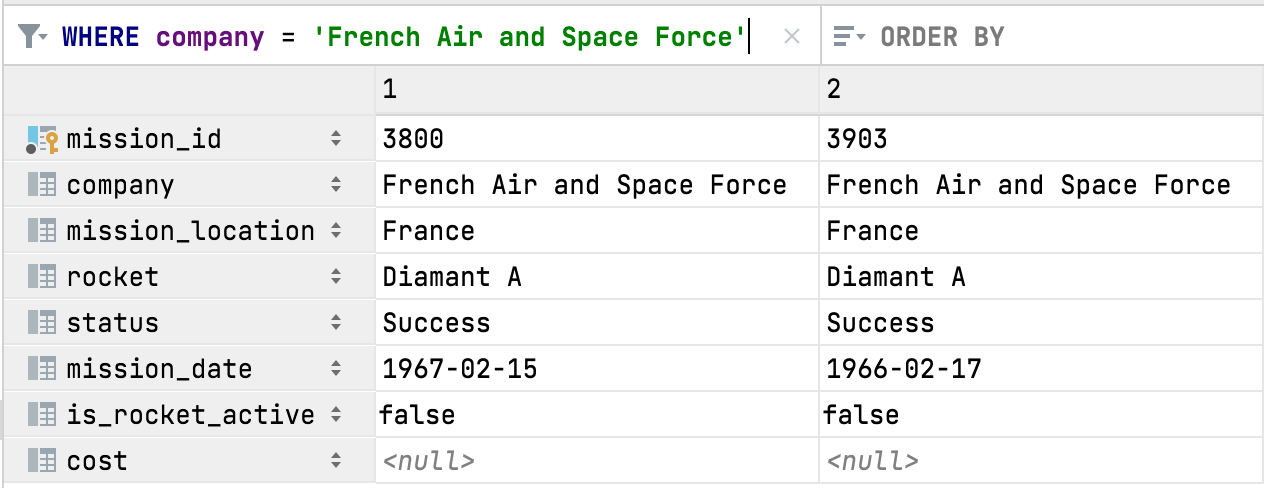
After googling we got "French Air and Space Force" corresponding to this value.

We replace this value with english translation to correct errors in encoding:

update mission

set company = 'French Air and Space Force'

where mission\_id in (select mission\_id from mission where company = 'Arm??e de l''Air');



Check mission location for wrong values:

select distinct mission\_location from mission;

No wrong values were found !

Check rocket names for wrong values:

select distinct rocket from mission;

No wrong values were found !

Check cost for wrong values:

select \* from mission where cost is not null

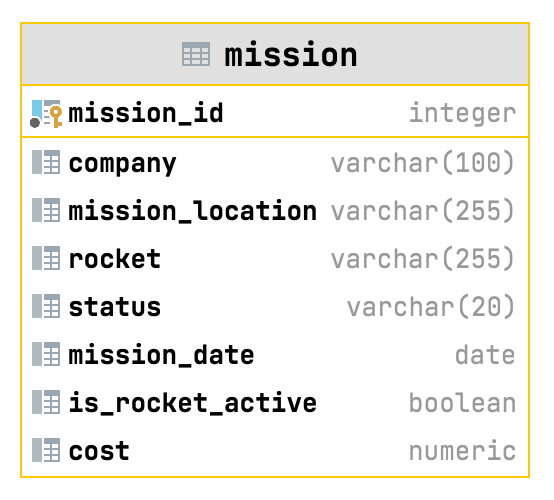
order by cost desc;

No wrong values were found !

## data Model

**Before splitting into fact and dimension tables:**

Lets see ERD of our dataset before splitting table into facts and dimensions:



**Splitting into fact and dimension tables:**

Splitting step is repetitive and tedious. For this I have created a function to automate the process of splitting columns into separate dimensions tables:

-- FUNCTION THAT CREATES DIMENSION TABLE FOR SPECIFIED COLUMN AND MODIFIES FACTS TABLE BY CREATING FK REFERENCE

-- INPUT:

-- ptable -> MAIN TABLE NAME

-- pcolumn -> COLUMN THAT WE WANT TO MOVE INTO SEPARATE TABLE

-- pnewtablename -> NAME OF NEW TABLE WE WANT TO CREATE

-- OUTPUT:

-- RETURNS SUCCESS MESSAGE OR ERROR MESSAGE.

CREATE OR REPLACE FUNCTION generate\_table\_for\_column(IN ptable VARCHAR, IN pcolumn VARCHAR, IN pnewtablename VARCHAR)

RETURNS TEXT

LANGUAGE plpgsql

AS

$$

DECLARE

col\_type text;

new\_table\_pk\_name text;

begin

ptable := lower(ptable);

pcolumn := lower(pcolumn);

pnewtablename := lower(pnewtablename);

-- FIND DATA TYPE OF TARGET COLUMN:

EXECUTE format('SELECT pg\_typeof(%I) FROM %I LIMIT 1', pcolumn, ptable) INTO col\_type;

-- CREATE COLUMN NAME FOR PK OF NEW TABLE:

SELECT pnewtablename || '\_id' INTO new\_table\_pk\_name;

-- CREATE NEW TABLE FOR TARGET COLUMN WITH PK AND COLUMN FOR VALUES:

EXECUTE format('CREATE TABLE %I( %I INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY, %I %s NOT NULL, CREATED\_AT DATE NOT NULL

DEFAULT CURRENT\_DATE)', pnewtablename, new\_table\_pk\_name, pcolumn, col\_type);

-- FILL NEW TABLE WITH DATA FROM SOURCE TABLE:

EXECUTE format('INSERT INTO %I (%I) SELECT DISTINCT %I FROM %I', pnewtablename, pcolumn, pcolumn, ptable);

-- ADD COLUMN FOR FOREIGN KEY RELATIONSHIP TO REFERENCE ID OF VALUE IN NEWLY CREATED TABLE:

EXECUTE format('ALTER TABLE %I ADD COLUMN %I INT', ptable, new\_table\_pk\_name);

-- FILL FK OF SOURCE TABLE COLUMN WITH PK VALUES FROM DIMENSION TABLE:

EXECUTE 'UPDATE '||quote\_ident(ptable)||' t1 SET '||quote\_ident(new\_table\_pk\_name)||' =

t2.'||quote\_ident(new\_table\_pk\_name)||' FROM '||quote\_ident(pnewtablename)||' t2 WHERE t1.'||quote\_ident(pcolumn)||' =

t2.'||quote\_ident(pcolumn);

-- ADD FOREIGN KEY CONSTRAINT:

EXECUTE format('ALTER TABLE %I ADD CONSTRAINT %s FOREIGN KEY (%I) REFERENCES %I(%I) ON UPDATE CASCADE ON DELETE RESTRICT',

ptable, ptable||'\_'||new\_table\_pk\_name||'\_fk', new\_table\_pk\_name, pnewtablename, new\_table\_pk\_name);

-- CREATE INDEX ON NEW FK COLUMN:

EXECUTE format('CREATE INDEX %s ON %I(%I)', ptable||'\_'||new\_table\_pk\_name||'\_fk\_index', ptable, new\_table\_pk\_name);

-- DROP COLUMN FROM SOURCE TABLE THAT CONTAINED VALUES THAT WERE REPLACED BY ID'S OF THAT VALUES TAKEN FROM NEW TABLE:

EXECUTE format('ALTER TABLE %I DROP COLUMN %I', ptable, pcolumn);

-- RETURN MESSAGE:

RETURN 'TABLE '|| pnewtablename || ' WAS CREATED !';

-- IF ERROR RETURN ERROR MESSAGE:

EXCEPTION WHEN OTHERS THEN RETURN SQLERRM;

end;

$$;

Calling function for columns: company, mission\_location, rocket:

SELECT generate\_table\_for\_column('mission', 'company', 'company');

SELECT generate\_table\_for\_column('mission', 'mission\_location', 'mission\_location');

SELECT generate\_table\_for\_column('mission', 'rocket', 'rocket');

Moving is\_rocket\_active column into rocket table:

alter table rocket add column is\_rocket\_active boolean;

update rocket

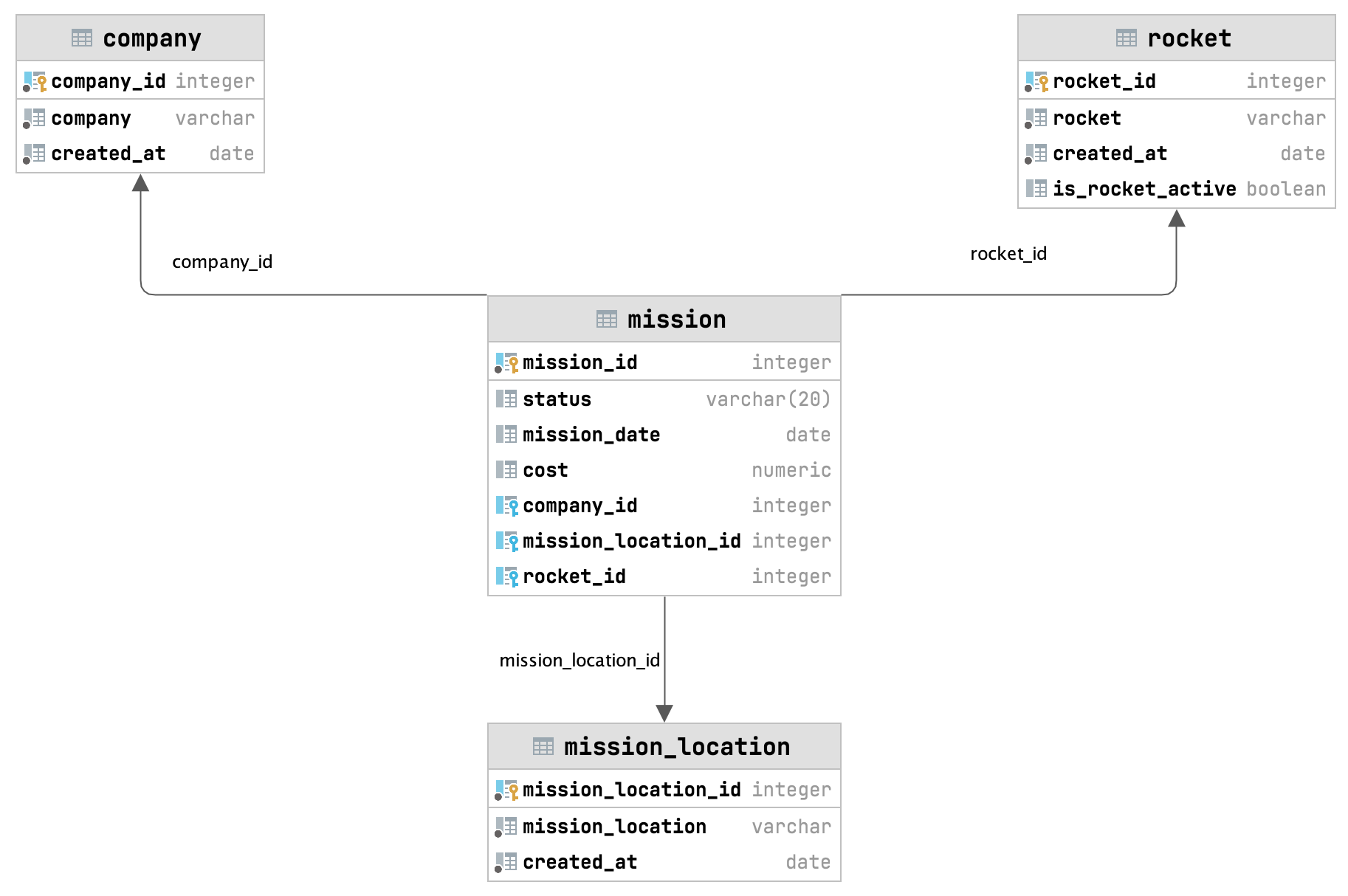
set is\_rocket\_active = mission.is\_rocket\_active

from mission

where rocket.rocket\_id = mission.rocket\_id;

alter table mission drop column is\_rocket\_active;

**Data model after splitting into fact and dimension tables:**



*STAR model of final dataset*

**company (dimension table)** – contains all companies owners of the rockets.

**rocket (dimension table )** – contains all rocket models in dataset and whether the rocket is active or not.

**mission\_location (dimension table)** – contains all location where rockets, where launched from.

**mission (fact table)** – contains all launch data with all references to dimension tables.

## TASK DESCRIPTION

Our telecom company based in Uzbekistan has a new satellite to increase coverage for our services. We have to send it into orbit as soon as possible with lowest possible price. We need to find a company that will get the contract for this mission. In order to find out the company we have to make analysis on provided data.

## data analysis

### Q1) IN WHICH MONTH THE LAUNCH HAS HIGHEST POSSIBILITY TO SUCCEED:

First thing we have to find the date for our mission. For that we need to find in which month the lauch has the highest probability to succeed.

#### **SQL QUERY**

# select CASE extract(month from m.mission\_date)

# WHEN 1 THEN 'January'

# WHEN 2 THEN 'February'

# WHEN 3 THEN 'March'

# WHEN 4 THEN 'April'

# WHEN 5 THEN 'May'

# WHEN 6 THEN 'June'

# WHEN 7 THEN 'July'

# WHEN 8 THEN 'August'

# WHEN 9 THEN 'September'

# WHEN 10 THEN 'October'

# WHEN 11 THEN 'November'

# WHEN 12 THEN 'December'

# END as month,

# count(mission\_id) as total\_success\_missions,

# sum(count(mission\_id)) over () as tatal\_missions,

# round(count(mission\_id) / sum(count(mission\_id)) over () \* 100, 2) as percentage

# from mission m

# where m.status = 'Success'

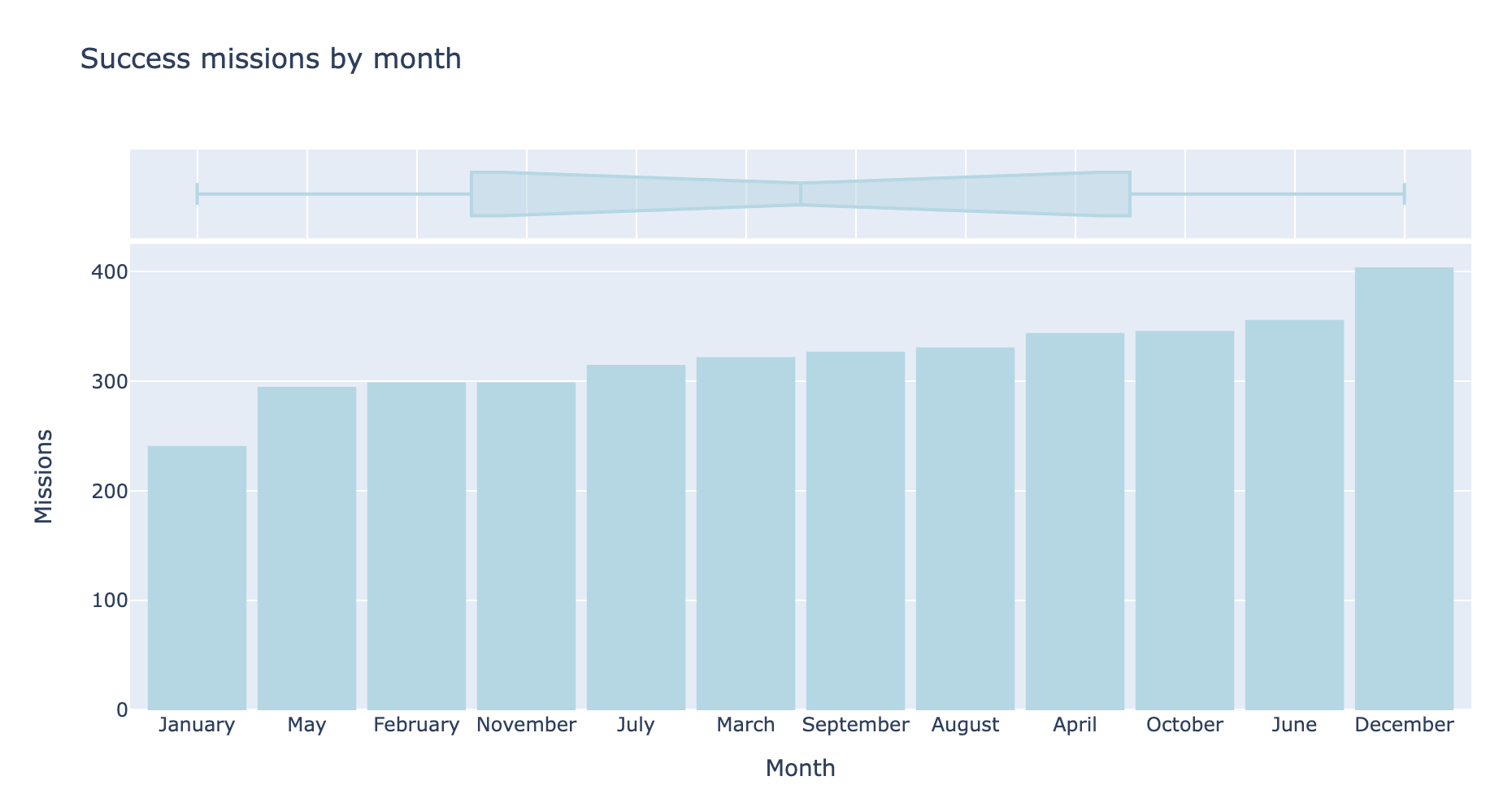
# group by extract(month from m.mission\_date)

# order by percentage asc;

#### **DATAFRAME**

### 

#### **PLOT**



#### **RESULT**

We can clearly see that we should schedule our launch to December as it will have the highest chance to succeed.

### Q2) TOP 10 COMPANIES BY NUMBER OF LAUNCHES:

We first can look for top companies by number of launches. More launches means more experience. From this analysis stage we can find our list of candidate companies.

#### **SQL QUERY**

# with cte\_launch\_data as (

# select company,

# status,

# total\_missions\_by\_status,

# total\_missions\_by\_company,

# dense\_rank()

# over (order by total\_missions\_by\_company desc, company

# groups between unbounded preceding and unbounded following) as rank

# from (

# select c.company,

# m.status,

# count(m.mission\_id) as total\_missions\_by\_status,

# sum(count(m.mission\_id)) over (partition by c.company) as total\_missions\_by\_company

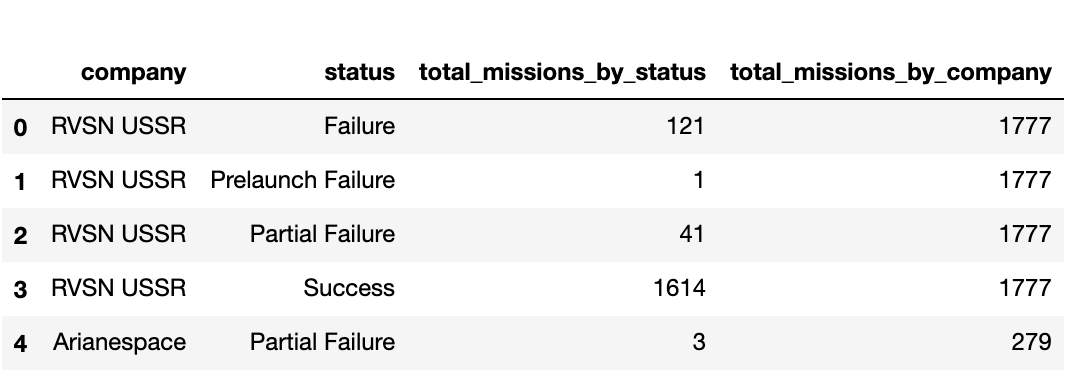
# from mission m

# inner join company c on m.company\_id = c.company\_id

# group by c.company, m.status) t

# ) select company, status, total\_missions\_by\_status, total\_missions\_by\_company from cte\_launch\_data where rank <= 10;

#### **DATAFRAME**



#### **PLOT**

### 

#### **RESULT**

We can see that we have 10 companies as RVSN USSR with 1777 launches (highest) and Martin Marietta with 114 launches (lowest).

We can further analyse these companies, however we know that USSR does not exists now and such RVSN. In addition some companies might not have operational rockets.  
  
We need a better method for finding our target companies.

### Q3) TOP 5 COMPANIES BY YEARLY INCREASE IN 5 YEAR PERIOD:

What we do here is that we first take companies that have active rockets ready for launch.  
We than calculate difference beetween success launches and failed launches to find absolute success factor of company.  
Lastly, we find yarly increase and it's total and take top 5 companies by amount of increase.

#### **SQL QUERY**

# with cte\_active\_companies as (

# select distinct c.company\_id, c.company

# from company c

# inner join mission m on c.company\_id = m.company\_id

# inner join rocket r on r.rocket\_id = m.rocket\_id

# where r.is\_rocket\_active = true

# ), cte\_yearly\_stats as (

# select extract(year from m.mission\_date) as year,

# ac.company,

# count(m.mission\_id) filter ( where m.status = 'Success' ) as succes\_missions,

# count(m.mission\_id) filter ( where m.status in ('Partial Failure', 'Failure', 'Prelaunch Failure') )

# as failed\_missions

# from mission m

# inner join cte\_active\_companies ac on ac.company\_id = m.company\_id

# where extract(year from mission\_date) between (select extract(year from max(mission\_date)) - 5 from mission)

# and (select extract(year from max(mission\_date)) from mission)

# group by extract(year from m.mission\_date), ac.company

# order by company

# ), cte\_change\_stats as (

# select year,

# company,

# succes\_missions - failed\_missions as positive\_mission\_count,

# (succes\_missions - failed\_missions) - first\_value(succes\_missions - failed\_missions)

# over (partition by company order by year) as change\_from\_first\_year

# from cte\_yearly\_stats

# ), cte\_total\_changee as (

# select year,

# company,

# positive\_mission\_count,

# change\_from\_first\_year,

# sum(change\_from\_first\_year) over (partition by company) as total\_increase

# from cte\_change\_stats

# ), cte\_ranking as (

# select cth.year,

# cth.company,

# cth.positive\_mission\_count,

# cth.change\_from\_first\_year,

# cth.total\_increase,

# dense\_rank() over (order by total\_increase desc, company

# groups between unbounded preceding and unbounded following) as rank

# from cte\_total\_changee cth

# ) select year,

# company,

# positive\_mission\_count,

# change\_from\_first\_year,

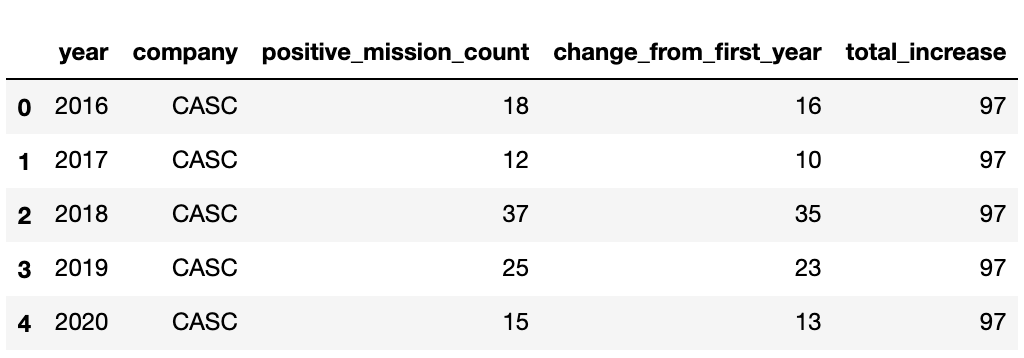
# total\_increase

# from cte\_ranking where rank <= 5

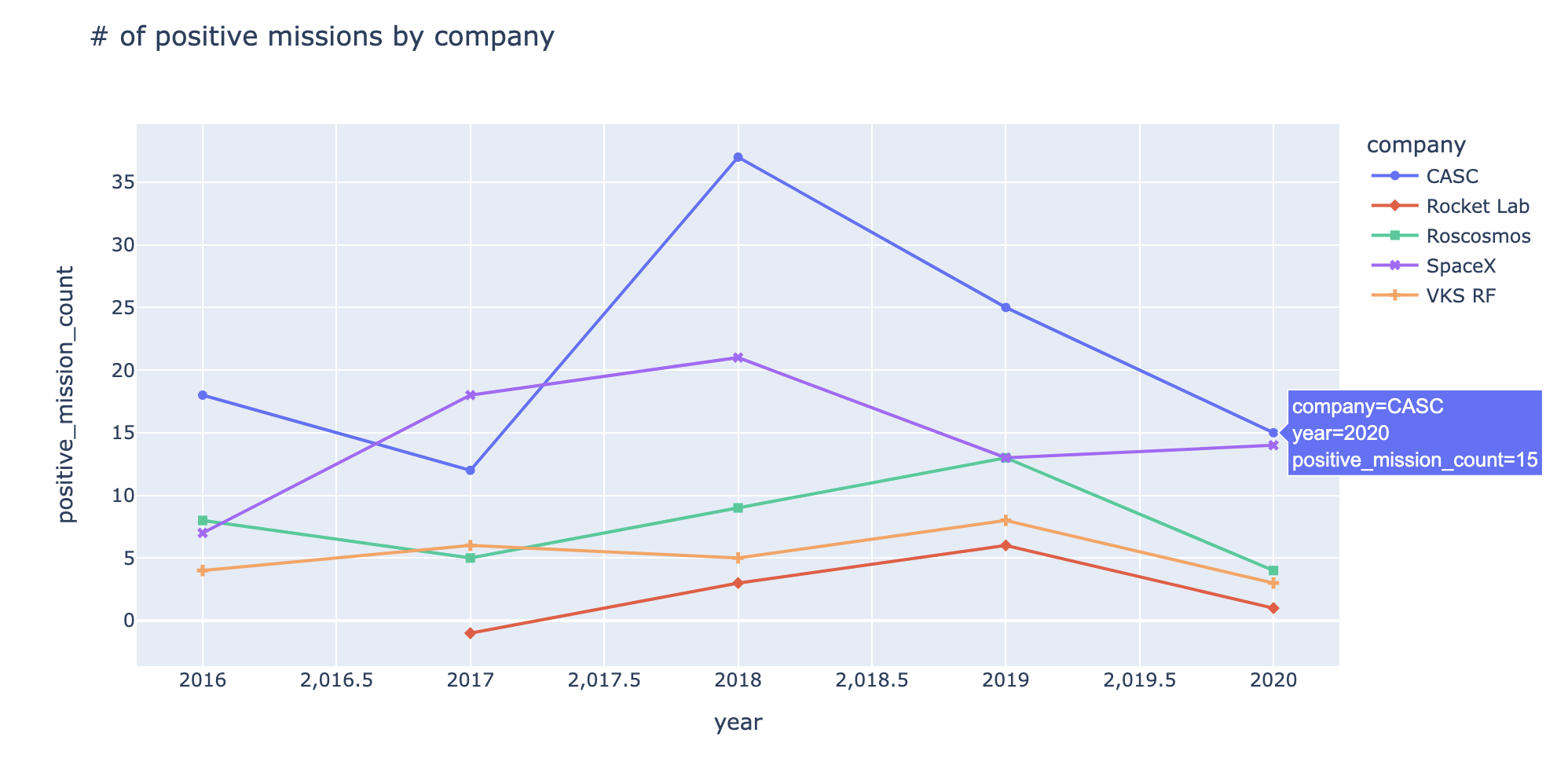
# and year between (select extract(year from max(mission\_date)) - 4 from mission)

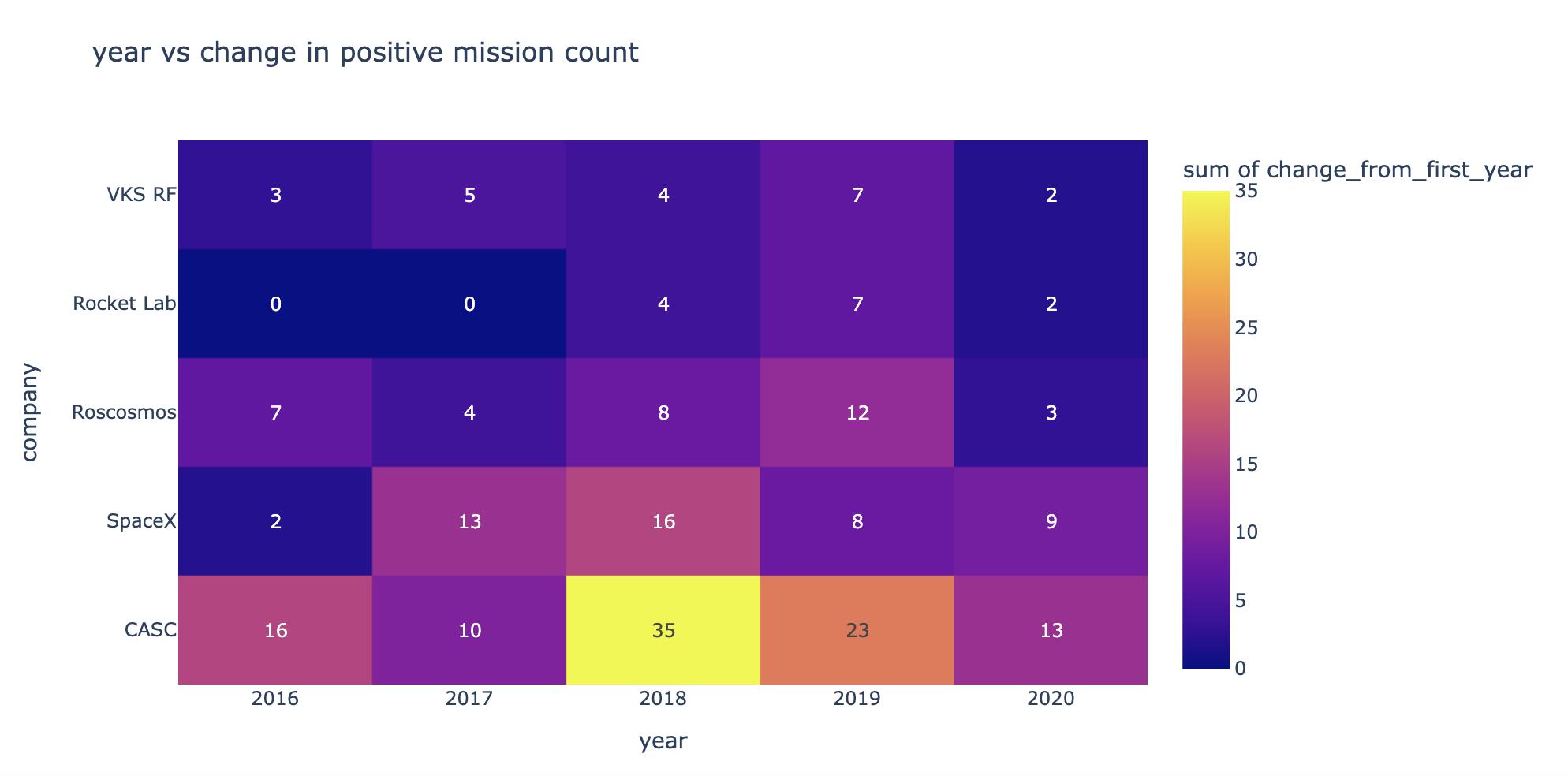
# and (select extract(year from max(mission\_date)) from mission) order by company, year;

#### **DATAFRAME**



#### **PLOT**





#### **RESULT**

From line chart we can see that CASC and Space X both have highest positive launch history. In addition, Space X in 2020 approaches CASC and has positive slope that indicates rise of performance.  
  
Furthermore, from Heatmap we see that again CASC and Space X both have the highest increase from 2016 in positive launch counts reaching 13 and 9 respectively.  
  
Based on results above, I have decided to consider only Space X and CASC for further analysis.

### Q4) MISSION COST FOR SpaceX AND CASC:

On this stage we have only 2 candidate companies left.  
As stated in task description, we have to send our payload as cheap as possible.  
In this analysis step, we will compare mission costs for both companies.

#### **SQL QUERY**

# CREATE VIEW company\_mission\_cost AS

# select c.company,

# ml.mission\_location,

# extract(year from mission\_date) as year,

# r.rocket,

# avg(m.cost) as average\_mission\_cost

# from mission m

# inner join company c on c.company\_id = m.company\_id

# inner join rocket r on m.rocket\_id = r.rocket\_id

# inner join mission\_location ml on m.mission\_location\_id = ml.mission\_location\_id

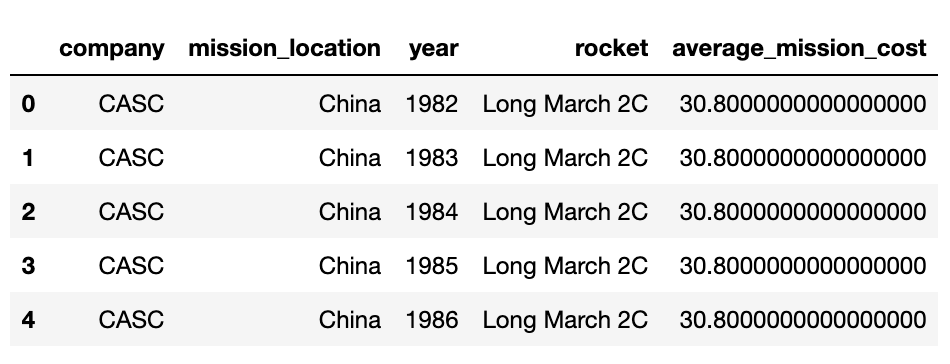
# where company in ('CASC', 'SpaceX')

# and m.cost is not null

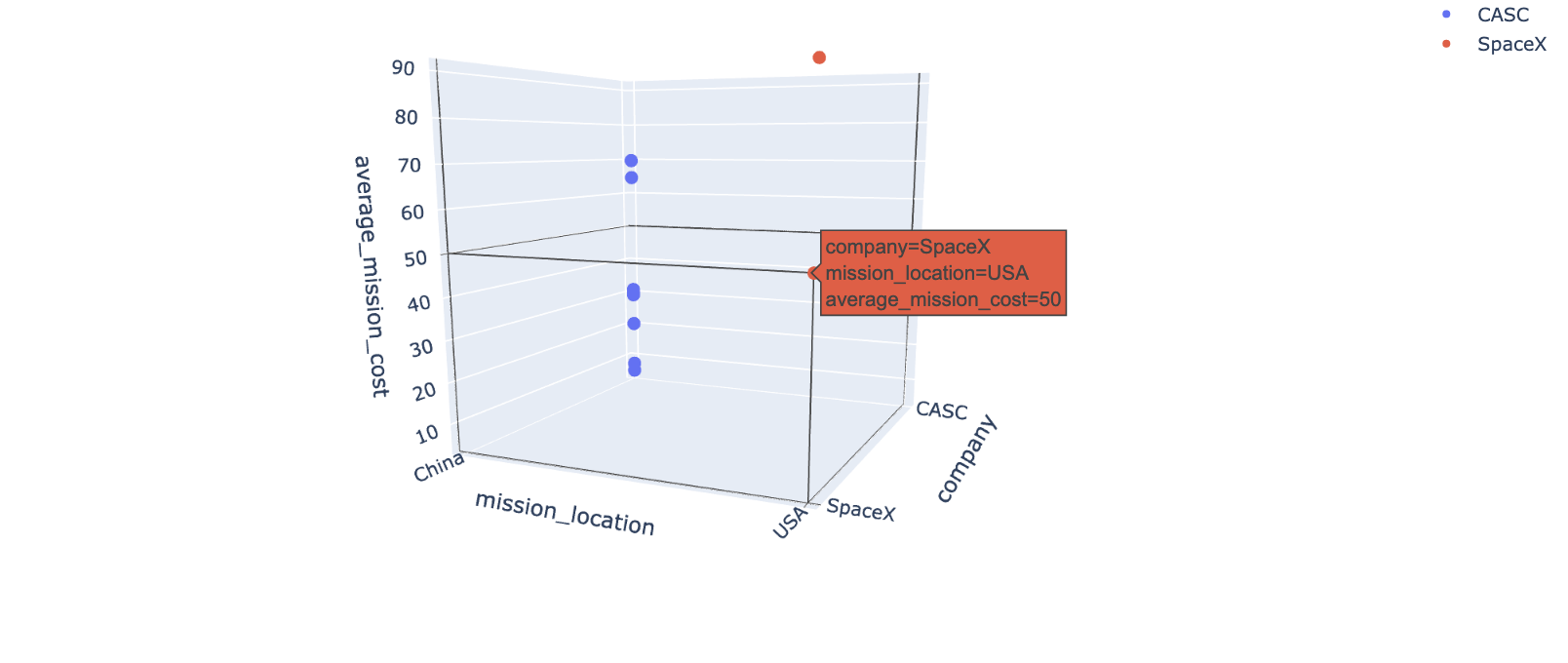
# and r.is\_rocket\_active = true

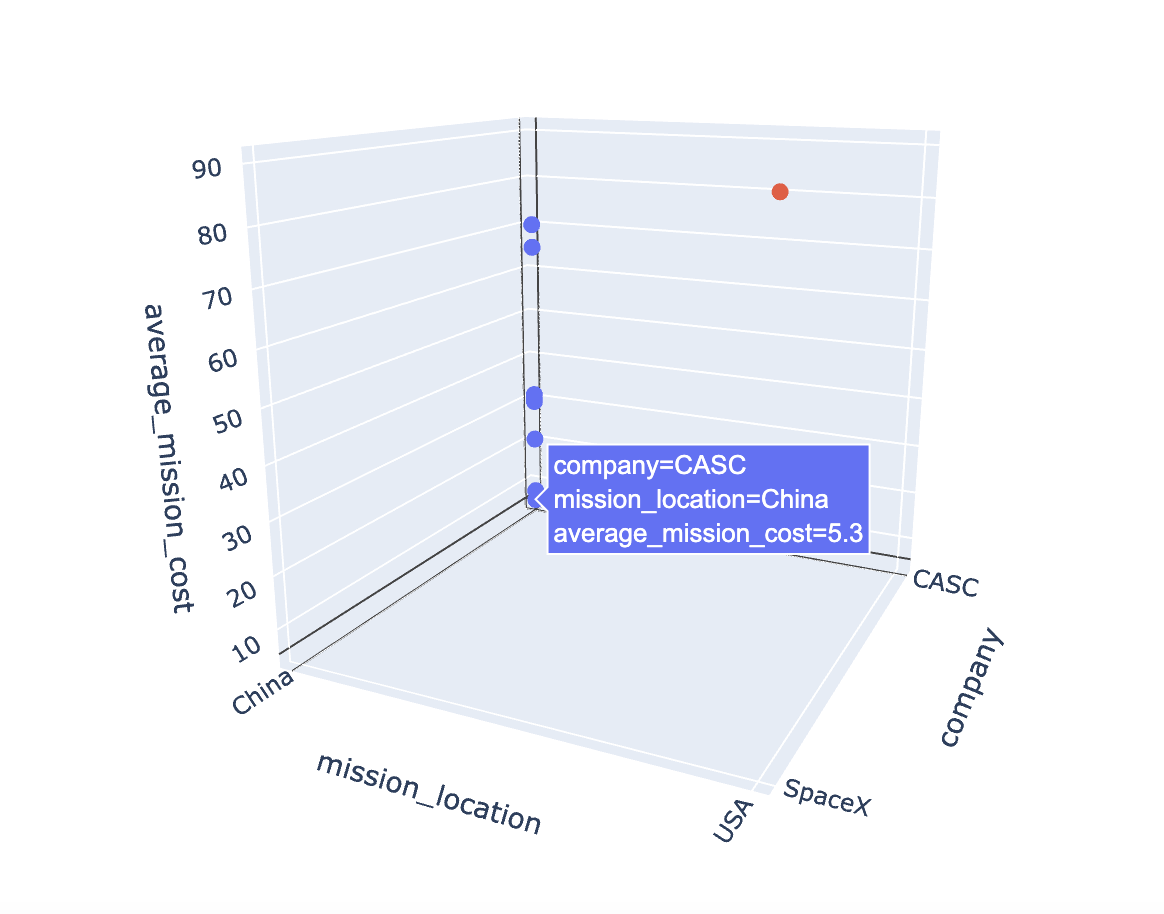
# group by c.company, ml.mission\_location, extract(year from mission\_date), r.rocket;

#### **DATAFRAME**

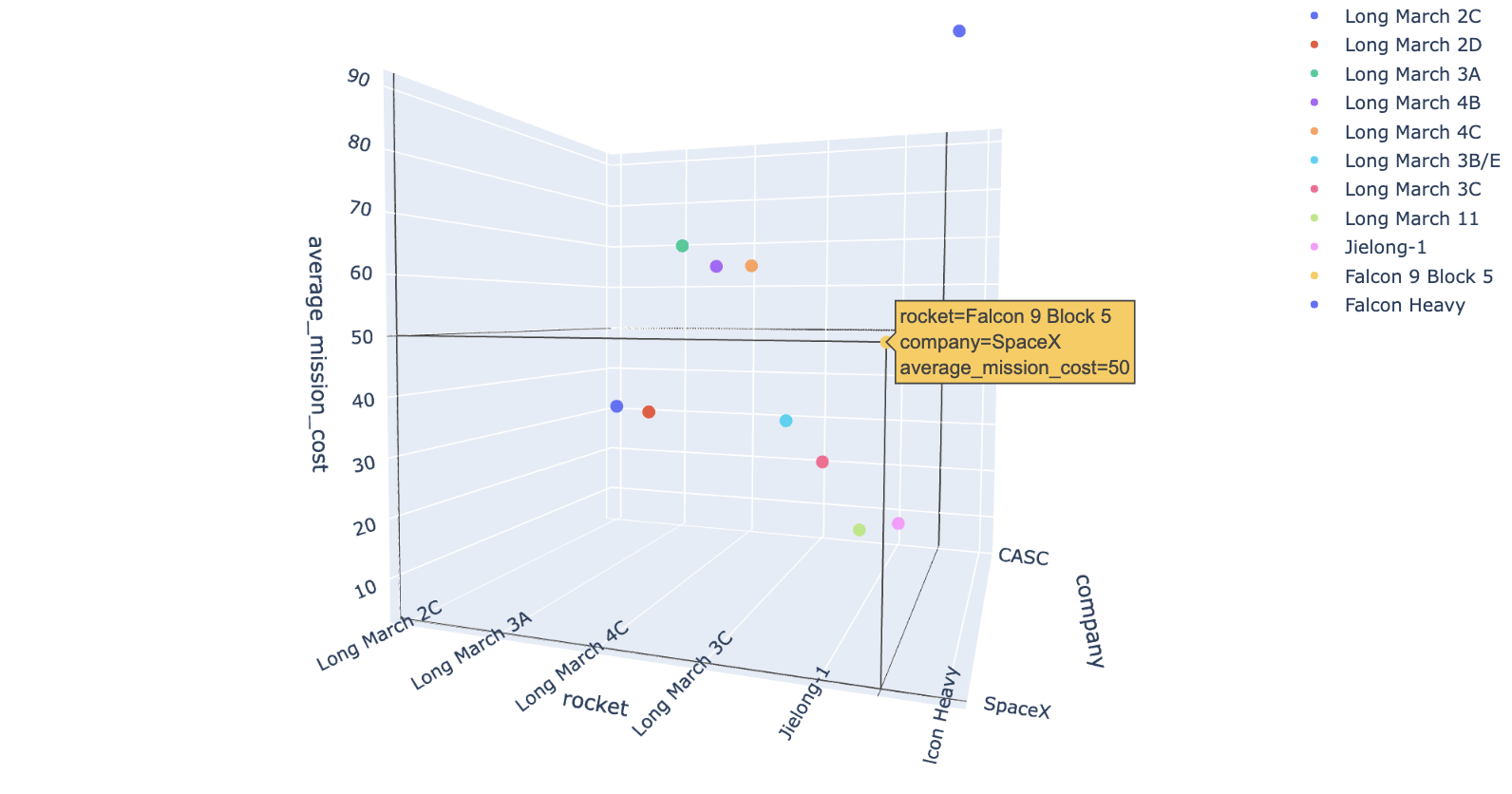


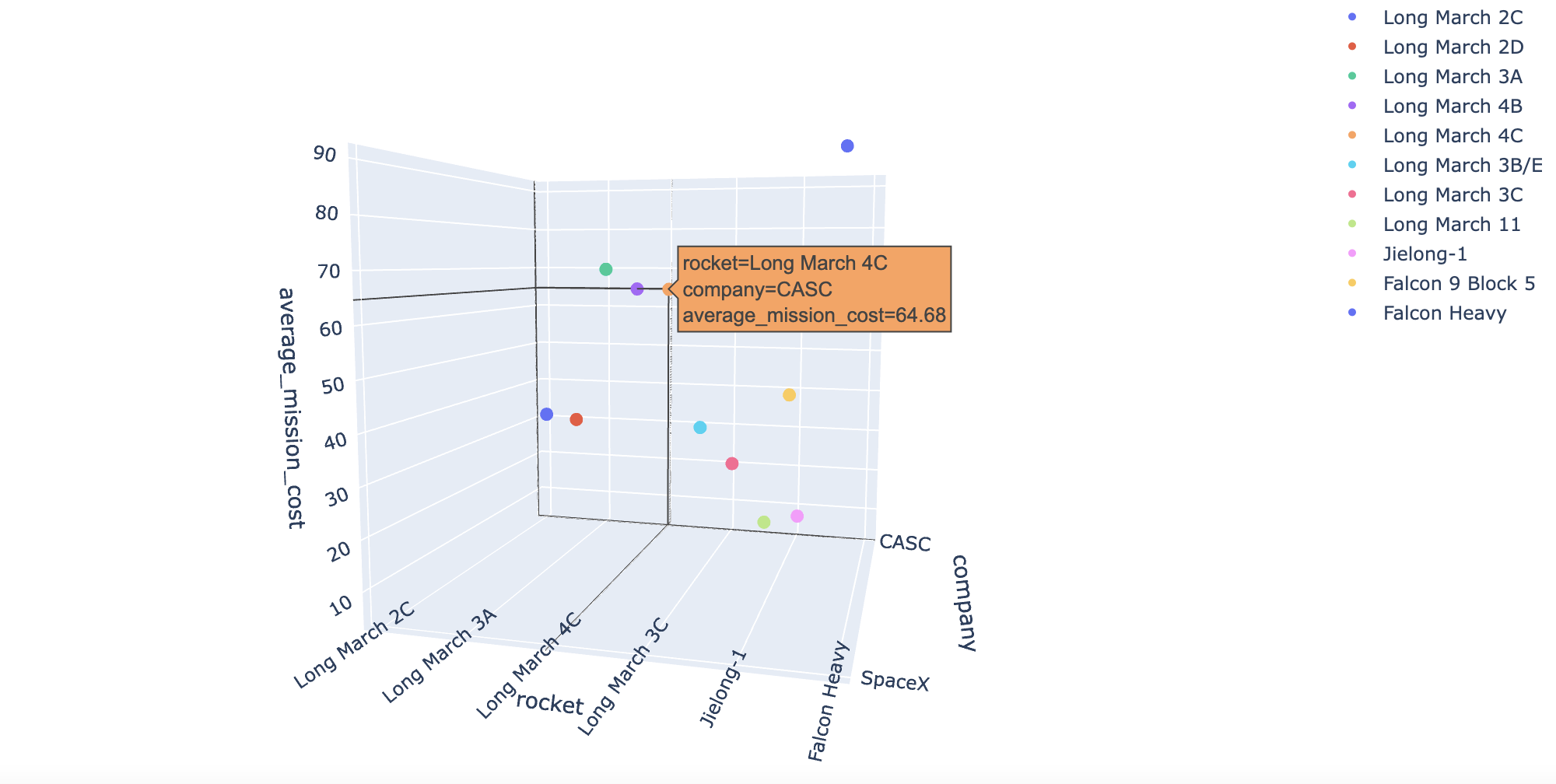
#### **PLOT by Location**





#### **PLOT by Rocket**





#### **RESULT**

From above 3d scatter plots we can see that we have two option in terms of country namely China and USA. As we are located in Uzbekistan, it seems that China is a good option for us in terms of logistics. It's costly to transfer our payload to USA.

However, In terms of rockets and overall cost, Falcon 9 by Space X is a best option in terms of price (50 millions) compared to Long March 4C and 4B (64.88 millions). I am comparing only these to rockets because they are all in the same class and are best suited for our task.

## SUMMARY

Based on above research and analysis, we pointed our 2 companies for our mission. They both have 2 advantages and 2 disadvantages in 4 main metrics.

However, our main goal was to complete the mission with lowest spendings as possible. Based on above analysis, the company that will get the contract is SpaceX.