

# Project Demo

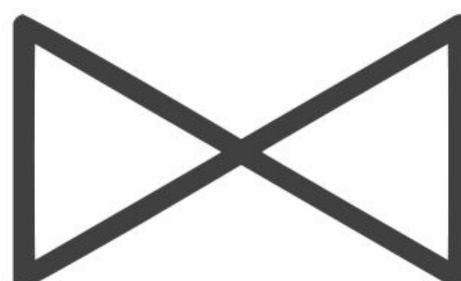
---

Samara Vassell, Isabella  
Baratta

Schema Schemers



# Introduction



Current standard SQL does not allow for users to compare aggregates or groups within a single query using GROUP BY. This project extends SQL to allow users to utilize groups in multiple ways within a single query to avoid unnecessary nested queries or joins.

# Agenda

01

Project Architecture

02

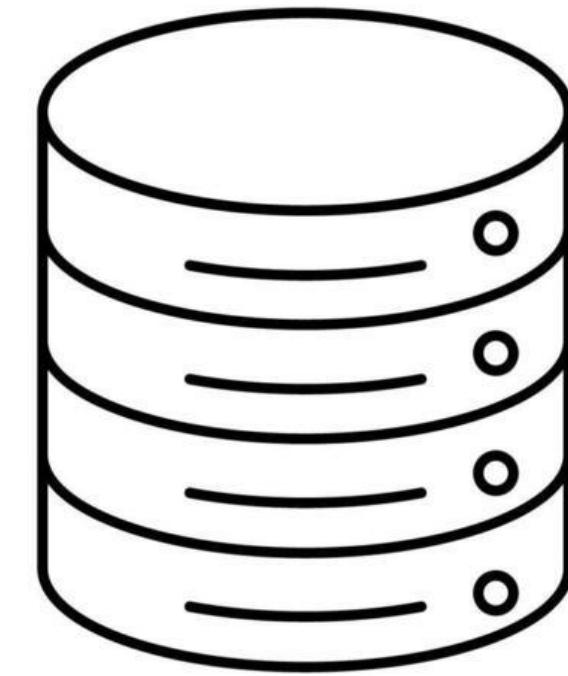
Tools and Methodology

03

Limitations

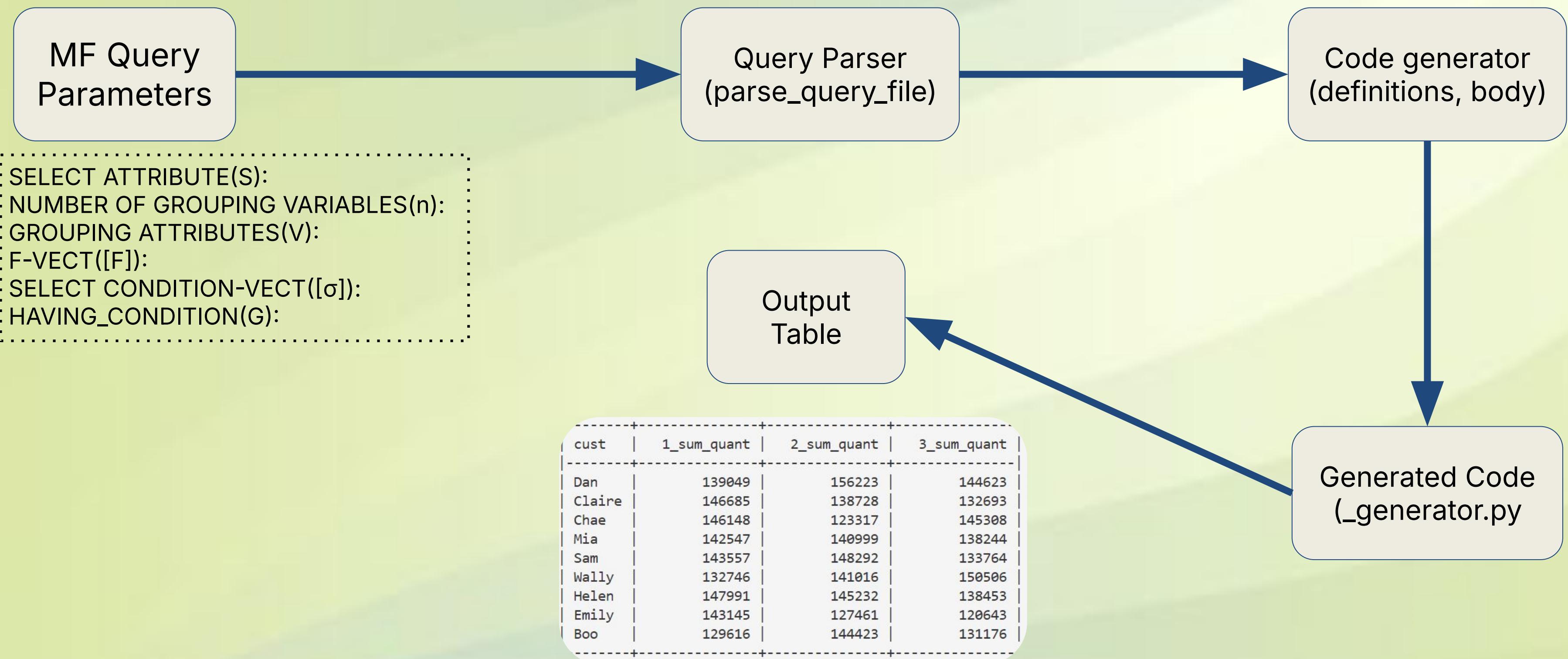
04

Demo



# Architecture

definitions = MFStructure, having +  
selection functions  
body = complete algorithm



# Tools and Methodology

Queries are presented by each different phi attribute.

Looking like:

SELECT ATTRIBUTE(S):

cust, 1\_sum\_quant

NUMBER OF GROUPING VARIABLES(V):

2

GROUPING ATTRIBUTES(V):

cust

F-VECT([F]):

1\_sum\_quant, 1\_avg\_quant

2\_avg\_quant

SELECT CONDITION-VECT( $\sigma$ ):

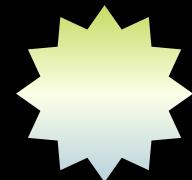
1.state='NY'

2.state='NJ'

HAVING\_CONDITION(G):

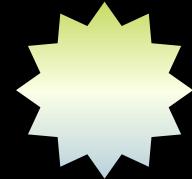
0

Our entire code is done within python and reads in an input text file if provided by user. Our code utilizes different python libraries/modules:



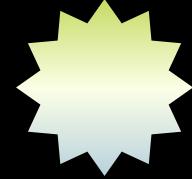
psycopg2

This allows python to work with the databases in postgres



Tabulate

Allows our queries to present a table for comparison purposes.



re Library

to differentiate the different aggregates and patterns

# Limitations

Dependent  
Aggregates that  
haven't occurred yet

There is no error checking  
for if a user presents  
`1.quant > 2_sum_quant` in  
the pred list

Does not support table  
joins

Takes the output of select  
statement for queries - no  
way to tell if a query needs  
a join unless input by user  
in future

Attributes for  
aggregates

Does not do auto-joins for  
queries such as "get dates  
of max quantities", where  
duplicates can occur

**DEMO!!!**

# Query 1: For each customer, get the sum of their purchases made in NY, NJ and CT

## Query

```
SELECT c.cust, ny.sum_ny, nj.sum_nj, ct.sum_ct
FROM (SELECT DISTINCT cust FROM sales) c
LEFT JOIN
  (SELECT cust, SUM(quant) AS sum_ny
   FROM sales WHERE state = 'NY'
   GROUP BY cust) ny ON c.cust = ny.cust
LEFT JOIN
  (SELECT cust, SUM(quant) AS sum_nj
   FROM sales WHERE state = 'NJ'
   GROUP BY cust) nj ON c.cust = nj.cust
LEFT JOIN
  (SELECT cust, SUM(quant) AS sum_ct
   FROM sales WHERE state = 'CT'
   GROUP BY cust) ct ON c.cust = ct.cust;
```

## Output (pgAdmin)

	cust character varying (20) 	sum_ny bigint 	sum_nj bigint 	sum_ct bigint 
1	Boo	129616	144423	131176
2	Chae	146148	123317	145308
3	Claire	146685	138728	132693
4	Dan	139049	156223	144623
5	Emily	143145	127461	120643
6	Helen	147991	145232	138453
7	Mia	142547	140999	138244
8	Sam	143557	148292	133764
9	Wally	132746	141016	150506

Query 2: For each state, get the max number of sales for Apples and Ham when max apples > max ham

## Query

```
WITH apple_sales AS (
    SELECT state, MAX(quant) AS max_apples
    FROM sales WHERE prod = 'Apple' GROUP BY state
),
ham_sales AS (
    SELECT state, MAX(quant) AS max_ham
    FROM sales WHERE prod = 'Ham' GROUP BY state
)
SELECT apple_sales.state, max_apples, max_ham
FROM apple_sales NATURAL JOIN ham_sales
WHERE max_apples > max_ham;
```

## Output

	state character (2)	max_apples integer	max_ham integer
1	CT	999	996
2	NJ	1000	997

Query 3: For each product, get count of sales in NY, average sales in NJ, and only show its value if the sales NJ average is > 500

## Query

```
WITH ny_counts AS (
    SELECT prod, COUNT(*) AS ny_sales_count
    FROM sales WHERE state = 'NY' GROUP BY prod
),
nj_avgs AS (
    SELECT prod, AVG(quant) AS nj_avg
    FROM sales WHERE state = 'NJ' GROUP BY prod
)
SELECT ny_counts.prod, ny_sales_count, nj_avg
FROM ny_counts NATURAL JOIN nj_avgs
WHERE nj_avgs.nj_avg > 500
```

## Output

	prod character varying (20)	ny_sales_count bigint	nj_avg numeric
1	Apple	265	512.3817427385892116
2	Ice	234	541.8278688524590164
3	Jelly	255	526.9915611814345992
4	Fish	246	503.6377358490566038
5	Butter	282	506.2263374485596708
6	Ham	254	504.0866141732283465
7	Cherry	257	507.2500000000000000

Query 4: Find the sum of quantity for each cust in NY, NJ, and CT where NJ and CT quantities are larger than NY average quantity

## Query

```

WITH q1 AS (
    SELECT cust, SUM(quant) as NY_sum_quant, AVG(quant) as NY_avg_quant
    FROM sales
    WHERE state = 'NY'
    GROUP BY cust
),
q2 AS (
    SELECT sales.cust, NY_sum_quant, SUM(sales.quant) as NJ_sum_quant
    FROM sales, q1
    WHERE sales.cust = q1.cust AND sales.state = 'NJ' AND sales.quant >
q1.NY_avg_quant
    GROUP BY sales.cust, q1.NY_sum_quant
)
SELECT sales.cust, q1.NY_sum_quant, q2.NJ_sum_quant, SUM(sales.quant) as
CT_sum_quant
FROM sales, q2, q1
WHERE sales.cust = q1.cust AND sales.cust = q2.cust AND sales.state = 'CT' AND
sales.quant > q1.NY_avg_quant
GROUP BY sales.cust, q1.NY_sum_quant, q2.NJ_sum_quant

```

## Output

	cust character varying (20)	ny_sum_quant bigint	nj_sum_quant bigint	ct_sum_quant bigint
1	Claire	146685	106872	99966
2	Sam	143557	107805	92634
3	Boo	129616	115044	99944
4	Wally	132746	115481	113030
5	Chae	146148	88523	110011
6	Emily	143145	94653	82369
7	Mia	142547	103862	104639
8	Helen	147991	111917	102754
9	Dan	139049	111308	112952

Query 5: For each customer and product find the max quantity for NY and the number of sales that are greater than NY's max for sales in NJ

## Query

```

WITH q1 AS (
    SELECT cust, prod, MAX(quant) as ny_max_quant
    FROM sales
    WHERE state = 'NY'
    GROUP BY cust, prod
)
SELECT sales.cust, sales.prod, q1.ny_max_quant,
COUNT(sales.quant)
FROM sales, q1
WHERE state = 'NJ' AND sales.cust = q1.cust AND
sales.prod = q1.prod AND sales.quant >
q1.ny_max_quant
GROUP BY sales.cust, sales.prod, q1.ny_max_quant

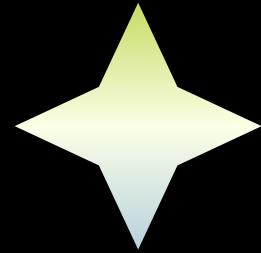
```

## Output

cust	prod	ny_max_quant	count
Boo	Apple	972	2
Boo	Ice	977	2
Boo	Jelly	964	1
Chae	Butter	972	1
Chae	Grapes	992	1
Chae	Ham	992	1
Chae	Jelly	914	4
Claire	Apple	929	2
Claire	Butter	980	1
Claire	Cherry	961	2
Claire	Ice	960	5
Claire	Jelly	914	3
Dan	Apple	937	1
Dan	Fish	986	1
Dan	Ham	967	1
Emily	Butter	950	1
Emily	Cherry	923	1
Emily	Eggs	976	1
Emily	Fish	973	2
Emily	Jelly	845	2

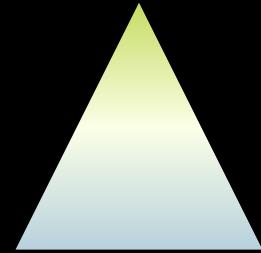
Helen	Butter	933	1
Helen	Cherry	965	4
Helen	Eggs	998	1
Helen	Fish	986	1
Helen	Grapes	988	1
Helen	Ham	897	2
Helen	Ice	979	1
Helen	Jelly	977	1
Mia	Cherry	949	4
Mia	Dates	855	3
Mia	Fish	937	2
Sam	Apple	888	7
Sam	Butter	972	2
Sam	Dates	988	1
Sam	Fish	930	1
Sam	Grapes	943	2
Sam	Ice	993	1
Wally	Butter	974	3
Wally	Dates	837	3
Wally	Eggs	934	4
Wally	Fish	982	1
Wally	Grapes	966	1
Wally	Ice	949	1

# DEMO



Finding or  
result here

Briefly discuss the  
outcome of your  
research or project.



Finding or  
result here

Briefly discuss the  
outcome of your  
research or project.

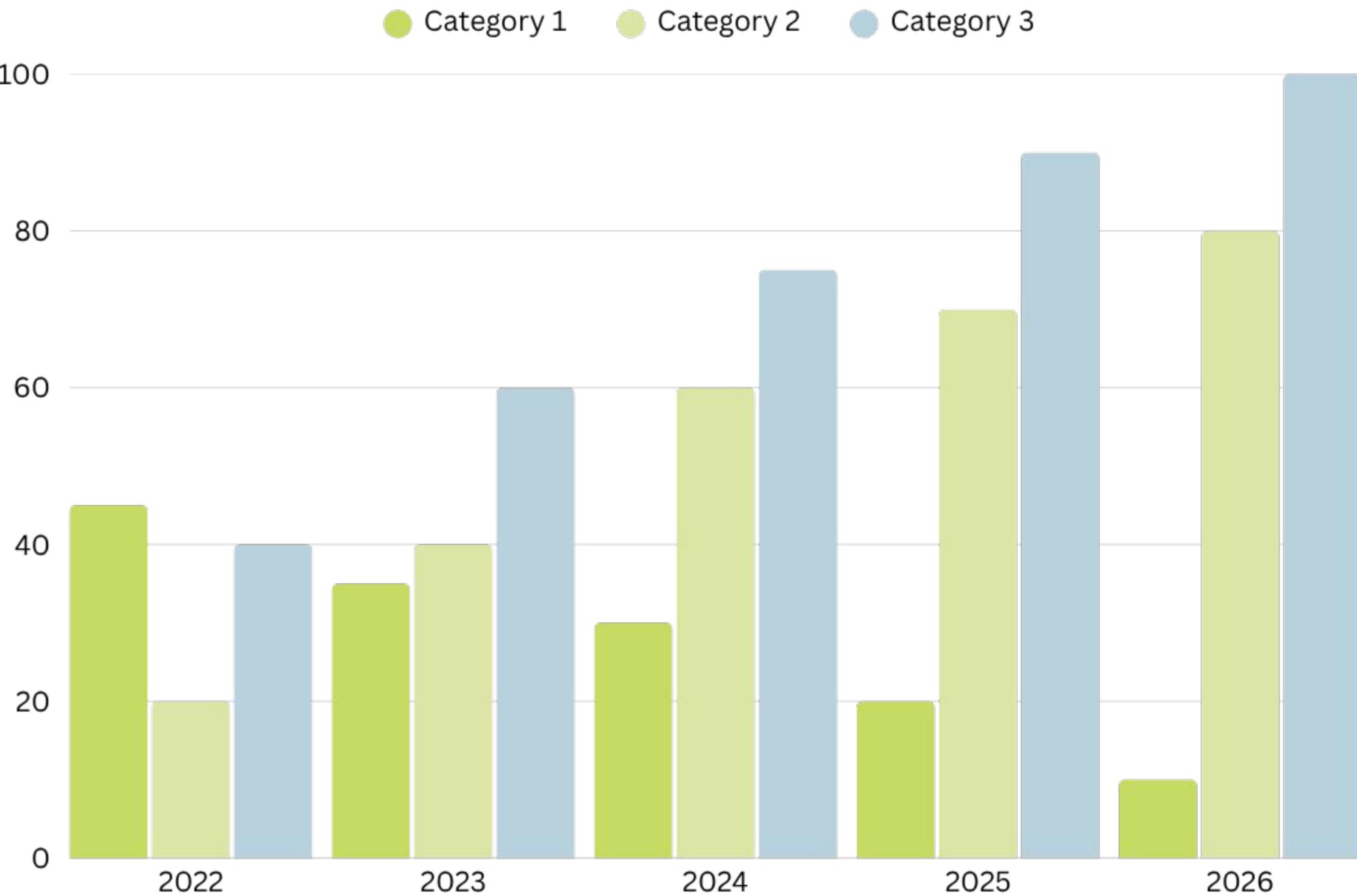


Finding or  
result here

Briefly discuss the  
outcome of your  
research or project.

# Chart title here

If applicable, use a chart or table to visualize your results or highlight comparisons. Use this section to briefly explain what the results mean.

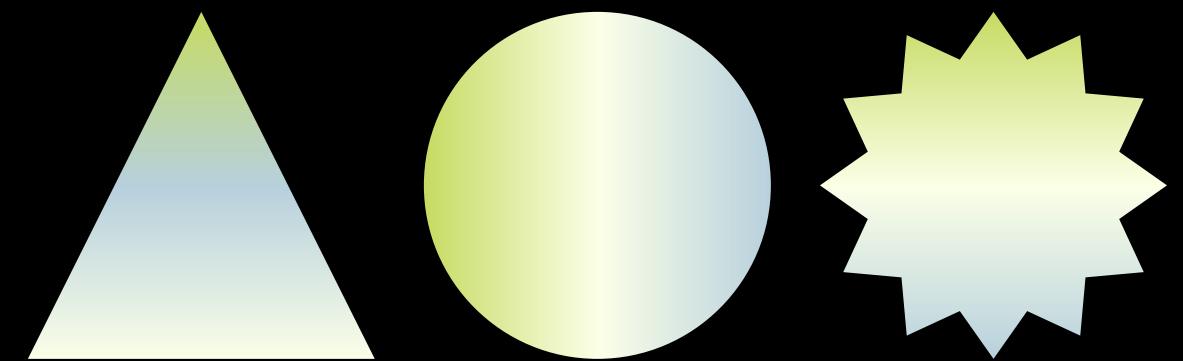


# Conclusion

Reflect on your project's impact and discuss how it relates to your initial goals or problem statement.



# Reach out



Email

[hello@reallygreatsite.com](mailto:hello@reallygreatsite.com)

Phone

123-456-7890

Social media

@reallygreatsite

Address

123 Anywhere St., Any City, ST

# Resource Page

Use these in your presentation. Delete or hide this page before presenting.

## Fonts

This presentation template  
uses the following free fonts:

### TITLES:

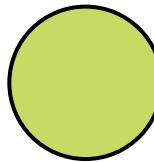
Raleway

### BODY TEXT:

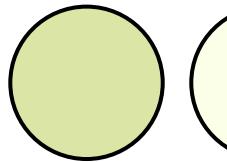
Inter

You can find these fonts online too.

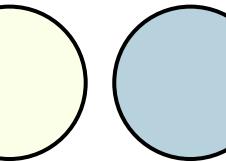
## Colors



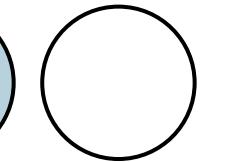
#C6DB63



#DBE6A6



#FBFFE7

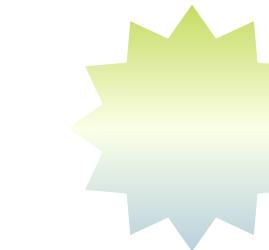
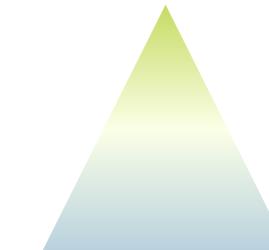
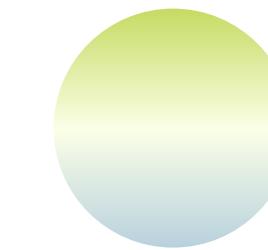
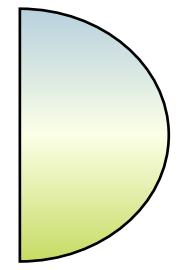


#B7D1DD



#FFFFFF

## Design Elements / Icons



# Credits

This presentation template is free for everyone to use, thanks to the following:

**Slides Carnival**

for this presentation template

Pexels, Pixabay,  
Sketchify

for the photos, graphics, and elements

Happy designing!