Ice Cream Sales Analysis (SQL + Python)

Name: Zalak Joshi
Purpose: Analyze ice cream sales dataset to find key insights, define KPIs, and suggest strategies for growth.

---

1. Setup

- Load libraries
- Connect to SQLite database
- Preview data

```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving analyst_exercise.db to analyst_exercise (2).db
```

```
import os
os.listdir()

import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


# connect to db
conn = sqlite3.connect('analyst_exercise (2).db')
```

```
pd.read_sql_query("SELECT name FROM sqlite_master WHERE type='table';", conn)
```

	name
0	ice_cream_sales

```
df = pd.read_sql_query("SELECT * FROM ice_cream_sales;", conn)
```

2. EDA

```
df.head()
```

	?	Customer	Date	Weekday	Flavor	Revenue	Profit	Margin
0		Drew Roulette	3/1/2018	5	Chocolate	$4.00	$3.00	75.00%
1		Don Frazon	3/2/2018	6	Vanilla	$4.00	$3.00	75.00%
2		Erin Davidson	3/2/2018	6	Vanilla	$4.00	$3.00	75.00%
3		Ted Price	3/2/2018	6	Chocolate	$4.00	$3.00	75.00%
4		Lila Stevens	3/3/2018	7	Cookie Dough	$5.00	$3.75	75.00%

```
df.shape
```

```
(86, 8)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86 entries, 0 to 85
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   ?         86 non-null     object
 1   Customer  86 non-null     object
 2   Date      86 non-null     object
 3   Weekday   86 non-null     object
 4   Flavor    86 non-null     object
 5   Revenue   86 non-null     object
 6   Profit    86 non-null     object
 7   Margin    86 non-null     object
dtypes: object(8)
memory usage: 5.5+ KB
```

Summary of Initial Exploration

- Dataset contains 8 columns and 86 records.
- All columns are stored as `object` type, requiring type conversion for analysis.
- No missing values are present, indicating clean data.
- One redundant column (`?`) was identified for removal.

```
df.describe()
```

	?	Customer	Date	Weekday	Flavor	Revenue	Profit	Margin
count	86	86	86	86	86	86	86	86
unique	1	12	27	6	5	4	4	2
top		Sierra Noonan	3/28/2018	7	Vanilla	$4.00	$3.00	75.00%
freq	86	10	14	20	26	38	38	72

Key Observations from `df.describe()`:
- There are **12 unique customers**, with **Sierra Noonan** having the highest number of transactions (10).
- There are **5 unique flavours**, with **Vanilla** being the most popular (26 orders).
- **March 28, 2018** saw the highest number of transactions (**14**).

```
df = df.drop(columns= ["?"])
```

```
df.head()
```

	Customer	Date	Weekday	Flavor	Revenue	Profit	Margin
0	Drew Roulette	3/1/2018	5	Chocolate	$4.00	$3.00	75.00%
1	Don Frazon	3/2/2018	6	Vanilla	$4.00	$3.00	75.00%
2	Erin Davidson	3/2/2018	6	Vanilla	$4.00	$3.00	75.00%
3	Ted Price	3/2/2018	6	Chocolate	$4.00	$3.00	75.00%
4	Lila Stevens	3/3/2018	7	Cookie Dough	$5.00	$3.75	75.00%

```
df['Revenue'] = df['Revenue'].str.replace('$', '', regex=False).astype(float)
df['Profit'] = df['Profit'].str.replace('$', '', regex=False).astype(float)
df['Margin'] = df['Margin'].str.replace('%', '', regex=False).astype(float)
```

```
df['Date'] = pd.to_datetime(df['Date'])
```

```
df['Weekday'] = df['Weekday'].astype(int)
```

```
df.dtypes
```

	0
Customer	object
Date	datetime64[ns]

	0
Weekday	int64
Flavor	object
Revenue	float64
Profit	float64
Margin	float64

3. Task 1: What insights can you find in the data set?

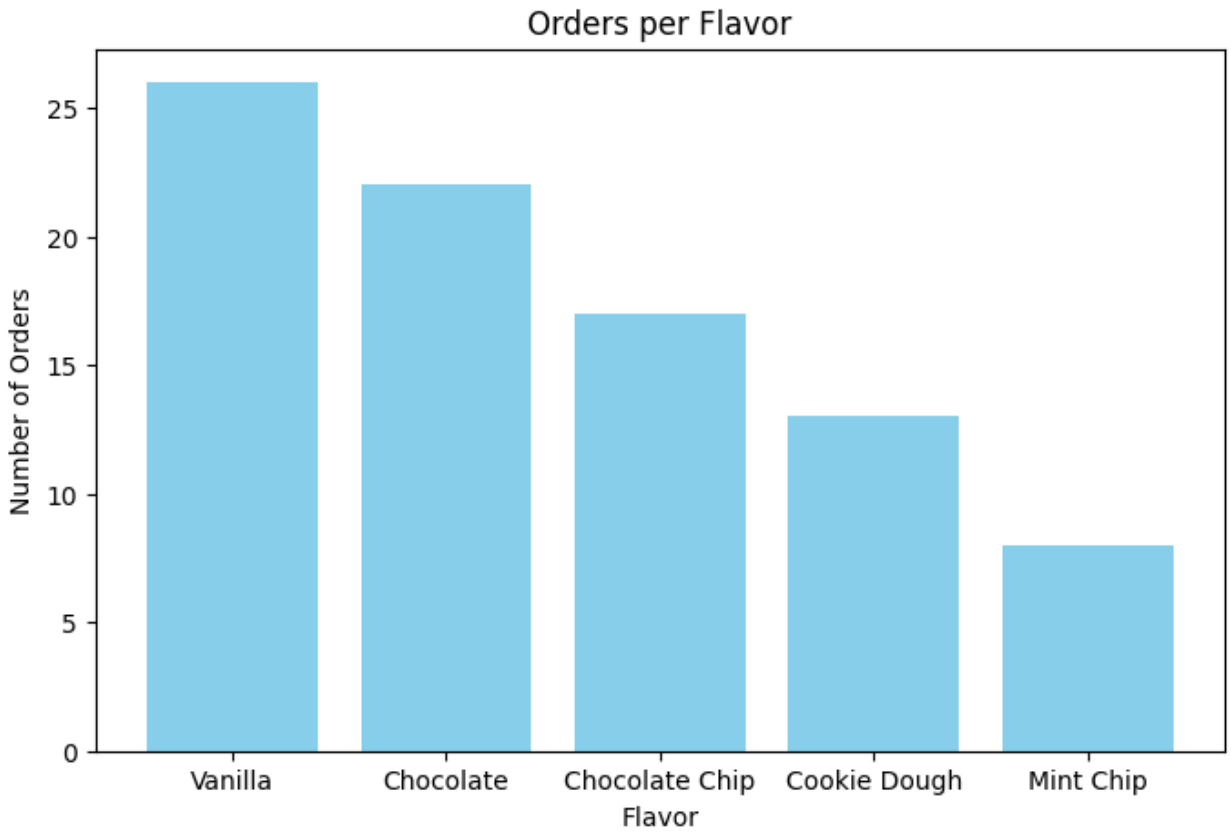
3.1 Flavor Popularity:

```
query = """
select flavor, count(*) as order_count
from ice_cream_sales
group by flavor
order by 2 desc
"""

df_flavor = pd.read_sql_query(query,conn)
df_flavor
```

	Flavor	order_count
0	Vanilla	26
1	Chocolate	22
2	Chocolate Chip	17
3	Cookie Dough	13
4	Mint Chip	8

```
plt.figure(figsize=(8,5))
plt.bar(df_flavor['Flavor'], df_flavor['order_count'], color='skyblue')
plt.title('Orders per Flavor')
plt.xlabel('Flavor')
plt.ylabel('Number of Orders')
plt.show()
```



Flavor Popularity Insights - **Vanilla** emerged as the **top-selling flavor** with **26 orders**, followed by **Chocolate (22 orders)**.
- **Mint Chip** was the **least popular** flavor.

3.2 Flavor-wise Revenue and Profit:

```
query = """
with cleaned as (
 select flavor,
       (cast(replace(revenue,'$','') as real)) as revenue_num,
       (cast(replace(profit,'$','') as real )) as profit_num
from ice_cream_sales
)
select flavor,
       sum(revenue_num) as Total_revenue,
       sum(profit_num) as Total_profit,
       round(sum(profit_num)*100.0/sum(revenue_num),2) as profit_margin_percent
from  cleaned
group by flavor
order by 2 desc;
"""
flavor_revenue_profit_df = pd.read_sql_query(query,conn)
```

```
flavor_revenue_profit_df
```

	flavor	Total_revenue	Total_profit	profit_margin_percent
0	Vanilla	90.0	64.00	71.11
1	Chocolate	82.0	60.00	73.17
2	Chocolate Chip	75.0	53.75	71.67
3	Cookie Dough	65.0	48.75	75.00
4	Mint Chip	40.0	30.00	75.00

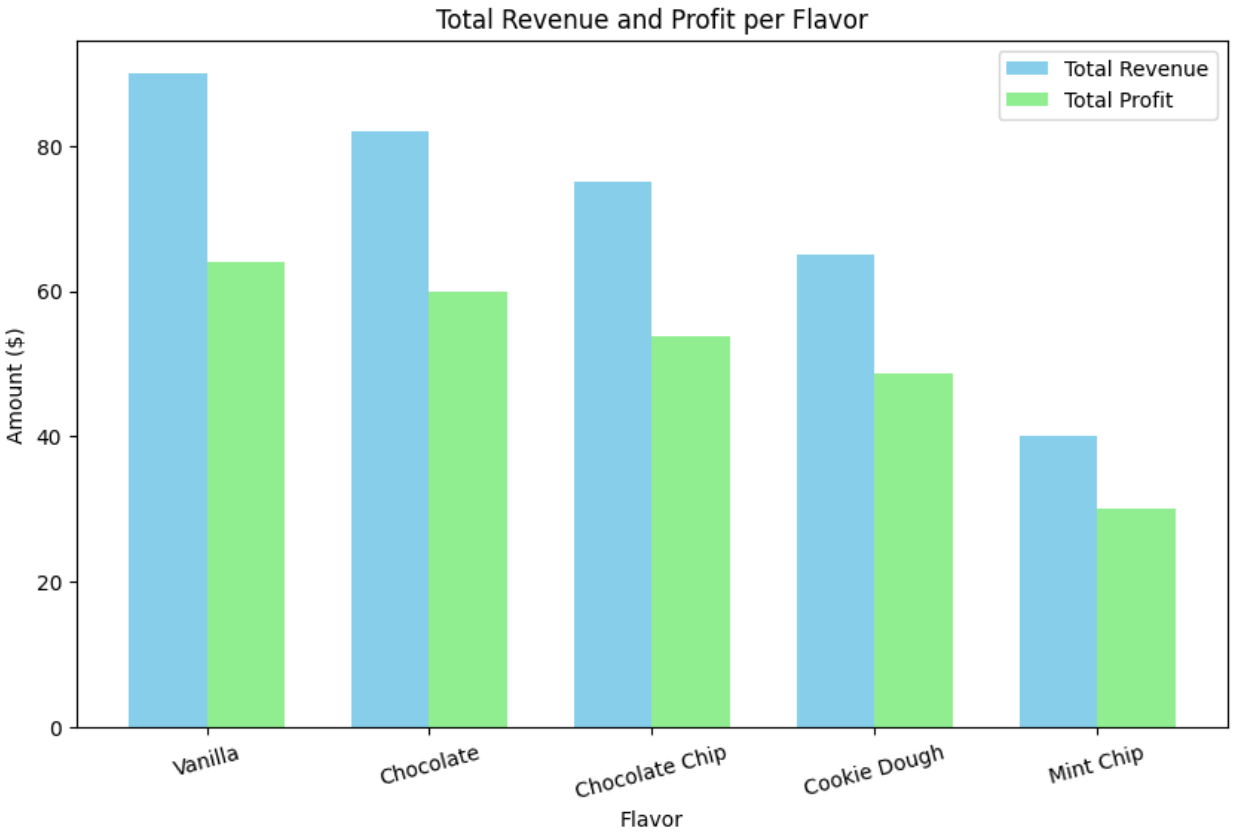
```
flavors = flavor_revenue_profit_df['flavor']
revenue = flavor_revenue_profit_df['Total_revenue']
profit = flavor_revenue_profit_df['Total_profit']

# Bar width & positions
x = np.arange(len(flavors))
width = 0.35

# Plot
plt.figure(figsize=(10,6))
plt.bar(x - width/2, revenue, width, label='Total Revenue', color='skyblue')
plt.bar(x + width/2, profit, width, label='Total Profit', color='lightgreen')


plt.xlabel('Flavor')
plt.ylabel('Amount ($)')
plt.title('Total Revenue and Profit per Flavor')
plt.xticks(x, flavors, rotation=15)
plt.legend()

plt.show()
```



Flavor-wise Revenue and Profit Insights - **Vanilla**: highest **total revenue $90** and **total profit $64**; **profit margin 71.11%** (lower than others). - **Chocolate**: second-highest **revenue $82** and **profit $60**; **margin 73.17%**. - **Cookie Dough**: **revenue $65**, **profit $48.75**; **margin 75.00%** (highest). - **Mint Chip**: **revenue $40**, **profit $30**; **margin 75.00%** (highest). - **Chocolate Chip**: mid-tier **revenue $75**, **profit $53.75**; **margin 71.67%**.

3.3 Top Customers by Revenue

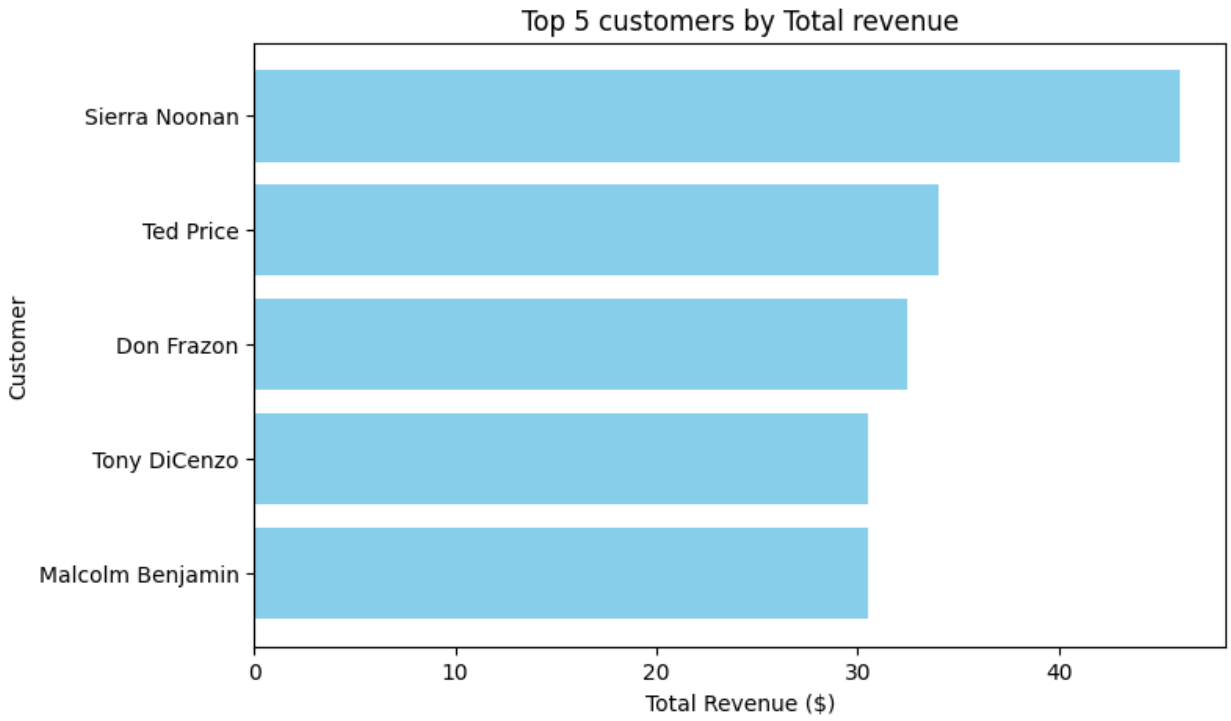
```
query= """
select Customer, sum(cast(replace(revenue,'$','') as real)) as Total_revenue
from ice_cream_sales
group by Customer
order by 2 desc
limit 5
"""

high_revenue_customer = pd.read_sql_query(query,conn)
high_revenue_customer
```

	Customer	Total_revenue
0	Sierra Noonan	46.0
1	Ted Price	34.0
2	Don Frazon	32.5
3	Tony DiCenzo	30.5
4	Malcolm Benjamin	30.5

```
customers = high_revenue_customer["Customer"]
revenue = high_revenue_customer["Total_revenue"]

plt.figure(figsize=(8,5))
plt.barh(customers,revenue, color = 'skyblue')
plt.xlabel('Total Revenue ($)')
plt.ylabel('Customer')
plt.title('Top 5 customers by Total revenue')
plt.gca().invert_yaxis()
```



Top Customers by Revenue Insights: - **Sierra Noonan** is the top customer, contributing **$46** in total revenue.
- **Ted Price** and **Don Frazon** follow with **$34** and **$32.5** respectively.
- Together, the top 5 customers contribute a significant portion of total revenue, making them a key target group for loyalty programs or upselling strategies.

3.4 Sales Trend Across Week

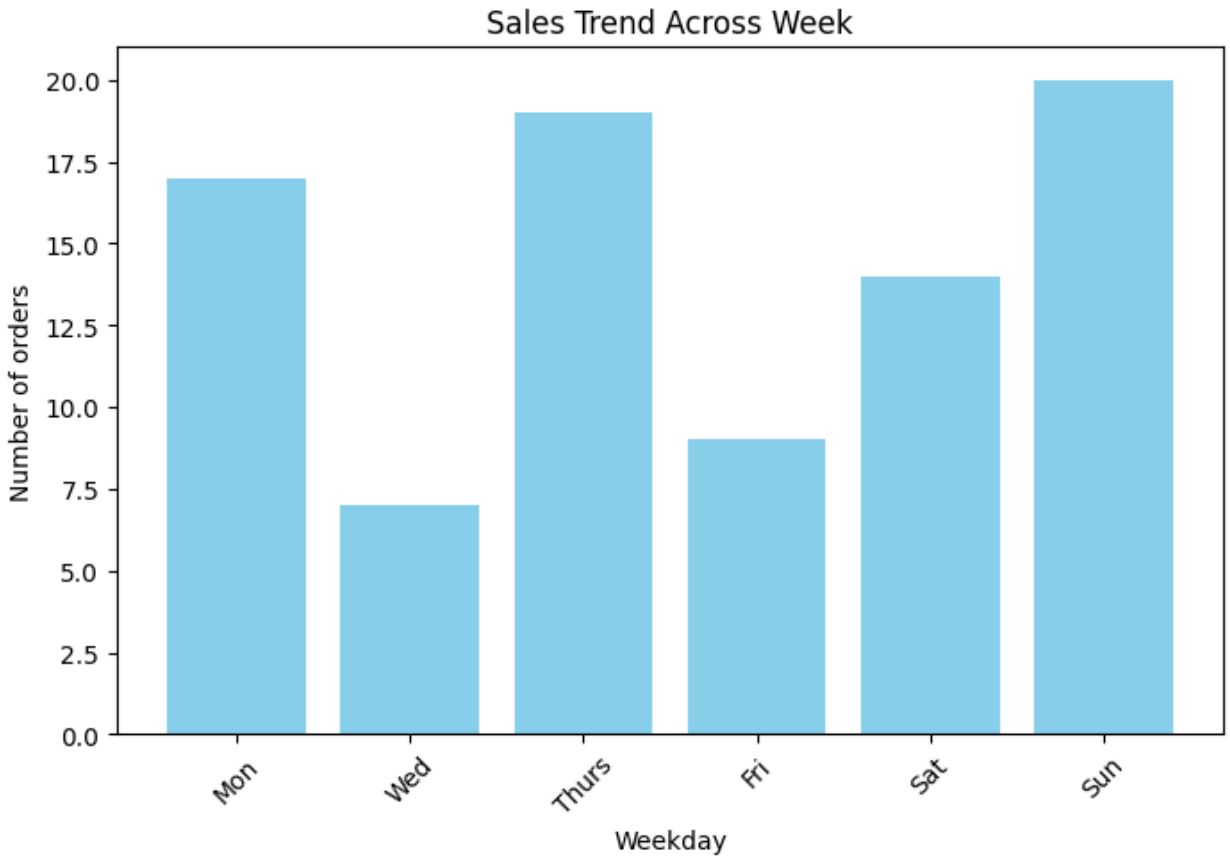
```
query= """

with Weekday_name as(
  select Customer, Weekday,
         case when Weekday = 1 then "Mon"
              when Weekday = 2 then "Tues"
              when Weekday = 3 then "Wed"
              when Weekday = 4 then "Thurs"
              when Weekday = 5 then "Fri"
              when Weekday = 6 then "Sat"
              when Weekday = 7 then "Sun"
         end as day
  from ice_cream_sales
)
select day,count(*) as order_count from Weekday_name
group by day
order by Weekday
"""

weekday_df = pd.read_sql_query(query,conn)
weekday_df
```

	day	order_count
0	Mon	17
1	Wed	7
2	Thurs	19
3	Fri	9
4	Sat	14
5	Sun	20

```
plt.figure(figsize=(8,5))
plt.bar(weekday_df['day'],weekday_df['order_count'], color = "skyblue")
plt.title("Sales Trend Across Week")
plt.xlabel("Weekday")
plt.ylabel("Number of orders")
plt.xticks(rotation= 45)
plt.show()
```



Weekday Sales Trend Insights - **Sunday** recorded the **highest number of orders (20)**.
- **Wednesday** had the **lowest sales (7)**.
- **Tuesday** had **no sales**, indicating the store was likely closed.
- Other weekdays, especially **Thursday (19)** and **Monday (17)**, also saw strong sales.

4. Task 2: KPI Selection

4.1 Total Revenue

```
query= """
select round(sum(cast(replace("Revenue","$","")as real)),2) as Total_Revenue
from ice_cream_sales
"""

pd.read_sql_query(query, conn)
```

	Total_Revenue
0	352.0

4.2 Total Profit

```
query= """
select round(sum(cast(replace("Profit","$","")as real)),2) as Total_Profit
from ice_cream_sales
"""

pd.read_sql_query(query, conn)
```

	Total_Profit
0	256.5

4.3 Average Order Value

```
query = """
select round(sum(cast(replace("Revenue","$","") as real))*1.0 /count(*),2) as
avg_order_value
from ice_cream_sales
"""

pd.read_sql_query(query, conn)
```

	avg_order_value
0	4.09

4.4 Week Over Week Profit

```
query = """
with date_split as (
select cast(substr(Date,length(Date)-3,4) as int) as y,
       cast(substr(Date,1,instr(Date,'/')-1) as int) as m,

cast(substr(Date,instr(Date,'/')+1,instr(substr(Date,instr(Date,'/')+1),'/')-1
)as int) as d,
       cast(replace(Profit,"$","")as real) as profit_num
from ice_cream_sales
),
join_date as (
```

```
select (cast(y AS text) || '-' ||
       substr('00'||m, -2, 2) || '-' ||
       substr('00'||d, -2, 2)) as formatted_date,
    profit_num
  from date_split
),
weekly as (
  select strftime('%Y-%W',formatted_date) as year_week,
         sum(profit_num) as weekly_profit
  from join_date
  group by year_week
),
final as
(
select year_week,weekly_profit,
       lag(weekly_profit)over(order by year_week) as prev_week_profit,
       round(weekly_profit - lag(weekly_profit)over(order by year_week)) *
100.0/nullif(lag(weekly_profit)over(order by year_week),0) as wow_growth_pct
from weekly
order by year_week
)
select wow_growth_pct from final order by  year_week desc limit 1;
"""
wow_df = pd.read_sql_query(query, conn)
wow_df
```

	wow_growth_pct
0	-10.928962

5. Task 4 Write SQL queries

5.1 Find the Top 2 flavours by Total Profit for Each Weekday

```
query = """
with req_data as (

  select weekday, Flavor,
         cast(replace(Profit,"$","") as real) as profit_num
  from ice_cream_sales
),
agg as (

  select weekday, Flavor,
         sum(profit_num) as total_profit
```

```
  from req_data
  group by weekday, Flavor
)
select Weekday, Flavor, round(total_profit,2) as Profit
from
(
select Weekday, Flavor, total_profit,
       rank() over(partition by Weekday order by total_profit desc ) as rnk
from agg
) temp
where rnk < 3;

"""

pd.read_sql_query(query,conn)
```

	Weekday	Flavor	Profit
0	1	Cookie Dough	15.00
1	1	Chocolate	12.00
2	1	Vanilla	12.00
3	3	Chocolate Chip	7.50
4	3	Chocolate	6.00
5	3	Vanilla	6.00
6	4	Cookie Dough	11.25
7	4	Vanilla	10.00
8	5	Chocolate	12.00
9	5	Vanilla	9.00
10	6	Chocolate Chip	15.00
11	6	Vanilla	12.00
12	7	Chocolate	15.00
13	7	Mint Chip	15.00
14	7	Vanilla	15.00

5.2 Find flavours Whose Sales Have Consistently Increased Week-Over-Week

```
query = """
with date_split as (
select flavor, cast(substr(Date,length(Date)-3,4) as int) as y,
       cast(substr(Date,1,instr(Date,'/')-1) as int) as m,

cast(substr(Date,instr(Date,'/')+1,instr(substr(Date,instr(Date,'/')+1),'/')-1
)as int) as d,
       cast(replace(Revenue,"$","")as real) as revenue_num
from ice_cream_sales
```

```
),
join_date as (
select flavor, (cast(y AS text) || '-' ||
       substr('00'||m, -2, 2) || '-' ||
       substr('00'||d, -2, 2)) as formatted_date,
    revenue_num
  from date_split
),
weekly as (
  select strftime('%Y-%W',formatted_date) as year_week, flavor,
         sum(revenue_num) as weekly_revenue
  from join_date
  group by flavor, year_week
),
prev_week as (
  select flavor, year_week,weekly_revenue,
        lag(weekly_revenue)over(partition by flavor order by year_week) as
prev_week_revenue
  from weekly
  ),
final as(
  select flavor,  case when prev_week_revenue is null then 0
                       when weekly_revenue > prev_week_revenue then 0
                       else 1
                  end as violated
  from prev_week
)
select flavor from final
group by flavor
having count(*) >=1
and sum(violated) = 0;
"""
pd.read_sql_query(query,conn)
```

	flavor
0	Mint Chip

5.3 Find the Day with the Highest Revenue for Each flavour

```
query = """
 with formatted_revenue as (
  select flavor, weekday,
         case when Weekday = 1 then "Mon"
              when Weekday = 2 then "Tues"
```

```
              when Weekday = 3 then "Wed"
              when Weekday = 4 then "Thurs"
              when Weekday = 5 then "Fri"
              when Weekday = 6 then "Sat"
              when Weekday = 7 then "Sun"
         end as day,
         sum(cast(replace(Revenue,"$","")as real)) as total_revenue
  from ice_cream_sales
  group by flavor, weekday
 )
select Flavor, day
from (
    select *, dense_rank()over(partition by Flavor order by total_revenue desc)
as rnk
    from formatted_revenue
     ) temp
where rnk = 1;
"""

pd.read_sql_query(query,conn)
```

	flavor	day
0	Chocolate	Sun
1	Chocolate Chip	Sat
2	Cookie Dough	Mon
3	Mint Chip	Sun
4	Vanilla	Sun

5.4 Calculate the Retention Rate: Customers Who Made Purchases in Both the First and Last Weeks

```
query = """
with date_split as (
select customer, cast(substr(Date,length(Date)-3,4) as int) as y,
       cast(substr(Date,1,instr(Date,'/')-1) as int) as m,

cast(substr(Date,instr(Date,'/')+1,instr(substr(Date,instr(Date,'/')+1),'/')-1
)as int) as d
from ice_cream_sales
),
join_date as (
select customer, (cast(y AS text) || '-' ||
       substr('00'||m, -2, 2) || '-' ||
       substr('00'||d, -2, 2)) as formatted_date
```

```
  from date_split
),
weekly as (
  select distinct(customer),strftime('%W',formatted_date) as week
  from join_date
),
first_week_cust as(
      select customer,week from weekly
      where week = (select min(week) from weekly)
),
last_week_cust as(
      select customer,week from weekly
      where week = (select max(week) from weekly)
),
returning_cust as(
  select f.customer
  from first_week_cust f
  join last_week_cust l
  on f.customer = l.customer
)
select (select count(*) from returning_cust) as ret_customer,
       (select count(*) from first_week_cust) as first_week_customer,
       round((select count(*) from returning_cust)* 100.0/ (select count(*) from
first_week_cust),2) as retention_rate_pct ;
"""
pd.read_sql_query(query,conn)
```

	ret_customer	first_week_customer	retention_rate_pct
0	8	9	88.89

6. Task 3 What suggestions can you give for increasing sales?

- **Mint Chip Promotions**: Offer targeted promotions on high footfall days (Sunday, Monday, Thursday) to boost its low sales volume without affecting margins.

- **Mid-week Discounts**: Provide limited-time discounts on popular flavors like Vanilla, Chocolate Chip, and Chocolate on **Wednesday and Friday**, which currently have the lowest sales.

- **Monthly Tuesday Campaigns**: Since the store remains closed on Tuesdays, once a month, use this day for **outreach and promotions outside the store** — e.g., setting up stalls, sampling in nearby areas, or distributing flyers to attract new customers.

- **Reverse Recent Decline**: Last two weeks showed a dip in sales. Launch a limited-time **combo or loyalty offer** to re-engage customers and restore momentum.

- **Focus on New Customer Acquisition**: With a strong **customer retention rate (88.89%)**, the focus should shift to **acquiring new customers** through local marketing, seasonal campaigns, and referral programs.

7. Conclusion

- The analysis highlighted clear patterns in flavour popularity, sales trends across weekdays, and customer behaviour.

- **Vanilla and Chocolate** are consistently top performers, while **Mint Chip** lags behind, offering room for growth through promotions.

- Sales are highest on **Sundays and Thursdays**, with mid-week days like **Wednesday and Friday** needing targeted interventions.

- The **customer retention rate (88.89%)** is strong, indicating loyalty is not a concern — efforts should focus on **acquiring new customers** and reversing the recent decline in weekly sales.

- By combining product-specific offers, mid-week discounts, and active marketing on Tuesdays, the store can **stabilize sales trends** and drive **sustainable growth**.