Part 1: SQL Implementation

For the part one, I created a random dataset using python first. Then, I loaded csv file of this dataset into my database using MySQL. Then, I performed a user funnel analysis using SQL queries. In addition to this, there are three more documents in the file. I included my Python and SQL codes and csv file of funnel analysis. Here

1- RandomDatasetwithPython.pdf
2- userevents1.sql
3- UserFunnelAnalysisMetrics.csv

1- My python codes (RandomDatasetwithPython.pdf):

```
*User Events Assesment*

Firstly I tried to create a random dataset named user_events with columns:
event_id (string)
user_id (string)
event_name (string) - possible values: 'PageView', 'Download', 'Install', 'Purchase'
platform (string) - possible values: ios and android
device_type (string)
timestamp (timestamp)

import pandas as pd
import numpy as np
from faker import Faker
import random

fake = Faker()

# Possible values for event_name and platform
event_names = ['PageView', 'Download', 'Install', 'Purchase']
platforms = ['ios', 'android']

# Generating random data
data = {
   'event_id': [fake.uuid4() for _ in range(100)],
   'user_id': [fake.uuid4() for _ in range(100)],
   'event_name': [random.choice(event_names) for _ in range(100)],
```

```
    'platform': [random.choice(platforms) for _ in range(100)],
    'device_type': [fake.word() for _ in range(100)],
    'timestamp': [fake.date_time_this_year() for _ in range(100)]
}


df_user_events = pd.DataFrame(data)

print(df_user_events.head())
```

```
                               event_id                               user_id  \
0  3324a8a0-8580-4c93-ba57-4690933fa44f  f6705b97-feac-4c85-8b11-0a5634188843
1  46599b55-f5b3-4345-b7f1-4da20944facf  74c24938-3fd1-4d38-bfb8-ef406168ec09
2  6af7d5ba-5c1e-4018-9e22-651fdfb8c1d0  87c4ec59-d7bc-4da8-b147-4104bfba865b
3  c895213f-1ce5-4523-bcdc-80ab49ab9aea  9f223767-b39f-47f3-8886-24d00312859d
4  30718df7-0feb-44fa-9dfa-010f121a3aaa  9b51699c-2520-46c6-86e3-3c909fee8164

  event_name platform device_type                  timestamp
0   PageView      ios      remain 2025-02-06 12:47:08.745290
1   PageView  android     himself 2025-02-03 11:37:28.081030
2   Purchase  android       write 2025-01-23 23:56:52.120980
3    Install      ios          PM 2025-01-30 04:24:40.706377
4   Purchase      ios       range 2025-01-08 10:14:08.798914
```

```
df_user_events.to_csv('user_events.csv', index=False, encoding='utf-8', sep=',')
df_user_events
```

	event_id	user_id	event_name	platform	device_type	timestamp
0	3324a8a0-8580-4c93-ba57-4690933fa44f	f6705b97-feac-4c85-8b11-0a5634188843	PageView	ios	remain	2025-02-06 12:47:08.745290
1	46599b55-f5b3-4345-b7f1-4da20944facf	74c24938-3fd1-4d38-bfb8-ef406168ec09	PageView	android	himself	2025-02-03 11:37:28.081030
2	6af7d5ba-5c1e-4018-9e22-651fdfb8c1d0	87c4ec59-d7bc-4da8-b147-4104bfba865b	Purchase	android	write	2025-01-23 23:56:52.120980
3	c895213f-1ce5-4523-bcdc-80ab49ab9aea	9f223767-b39f-47f3-8886-24d00312859d	Install	ios	PM	2025-01-30 04:24:40.706377
4	30718df7-0feb-44fa-9dfa-010f121a3aaa	9b51699c-2520-46c6-86e3-3c909fee8164	Purchase	ios	range	2025-01-08 10:14:08.798914
...	...	...	...	...	...	...

95	a0e712d3-f8d2-4399-818c-928b5bca9963	53ba56c1-4206-4304-b323-97b215987a3d	Purchase	android	exist	2025-01-21 14:49:20.437354
96	e3dc997f-e685-4c14-9c46-77dfcb523175	d43a5964-a59d-4d9e-b646-c6f3bc23423e	Purchase	ios	much	2025-02-15 01:00:10.810872
97	da5b513e-3558-4562-9418-b5ffc4568583	a1089aff-ca47-4925-aeb3-04967aea965f	Install	android	follow	2025-02-04 06:34:50.363219
98	207eb58b-f74d-4437-9663-3050b0215b30	8790d165-d964-4d18-aaeb-4efa1f238666	Install	android	hour	2025-01-02 06:15:14.241510
99	22b69516-ae03-434d-9dc8-3ff135a96c38	674e40a0-aa93-492b-8921-390268e4830e	Purchase	ios	may	2025-02-15 23:14:23.271926

100 rows × 6 columns

Then to make the dataset more logical, I divided the user event steps by realistic percentages with incresing number of users and events.

```
from faker import Faker
import random
from datetime import timedelta
import pandas as pd

fake = Faker()

# Possible values
event_names = ['PageView', 'Download', 'Install']
platforms = ['ios', 'android']
device_types = ['Phone', 'Tablet']

# The number of users
num_users = 1000  # Adjust as needed

# A list to store events
events = []

for _ in range(num_users):
    user_id = fake.uuid4()  # Unique user

    # 1: PageView (100% of users)
    events.append({
```

```
        'event_id': fake.uuid4(),
        'user_id': user_id,
        'event_name': 'PageView',
        'platform': random.choice(platforms),
        'device_type': random.choice(device_types),
        'timestamp': fake.date_time_this_year()
    })

    # 2: Download (80% chance)
    if random.random() < 0.8:
        events.append({
            'event_id': fake.uuid4(),
            'user_id': user_id,
            'event_name': 'Download',
            'platform': events[-1]['platform'],  # Same platform as PageView
            'device_type': events[-1]['device_type'],  # Same device type
            'timestamp': events[-1]['timestamp'] + timedelta(hours=random.randint(1, 72))  # Within 72 hours
        })

        # 3: Install (90% of those who downloaded)
        if random.random() < 0.9:
            events.append({
                'event_id': fake.uuid4(),
                'user_id': user_id,
                'event_name': 'Install',
                'platform': events[-1]['platform'],
                'device_type': events[-1]['device_type'],
                'timestamp': events[-1]['timestamp'] + timedelta(hours=random.randint(1, 72))
            })


df_user_events = pd.DataFrame(events)

df_user_events.to_csv("user_events1.csv", index=False)

print(df_user_events.head())
```

```
                             event_id                               user_id  \
0  d24efe9d-c04e-4eae-b1ef-42f030df35a7  019d87ce-756d-4a56-909f-dbaf0a5b5eb3
1  0c234e14-a01b-4d8c-bf70-a64cb5997e1e  019d87ce-756d-4a56-909f-dbaf0a5b5eb3
2  923aa092-a566-42ef-8e0a-77665f9e4bb0  019d87ce-756d-4a56-909f-dbaf0a5b5eb3
3  724935ea-012d-4a2c-85ff-15e0e790308e  98cedc91-5230-4b66-b7f2-edbffe8d8833
4  051392b2-e239-4a4a-9dad-4730977d6776  3ca4b9f4-b223-4080-8de7-fce445b39789

  event_name platform device_type                  timestamp
0   PageView      ios       Phone 2025-02-18 16:14:59.866918
1   Download      ios       Phone 2025-02-19 18:14:59.866918
2    Install      ios       Phone 2025-02-19 23:14:59.866918
3   PageView      ios       Phone 2025-01-11 15:41:35.254317
4   PageView      ios       Phone 2025-01-18 22:12:25.750521
```

```
df_user_events.to_csv('user_events1.csv', index=False, encoding='utf-8', sep=',')
df_user_events
```

	event_id	user_id	event_name	platform	device_type	timestamp
0	d24efe9d-c04e-4eae-b1ef-42f030df35a7	019d87ce-756d-4a56-909f-dbaf0a5b5eb3	PageView	ios	Phone	2025-02-18 16:14:59.866918
1	0c234e14-a01b-4d8c-bf70-a64cb5997e1e	019d87ce-756d-4a56-909f-dbaf0a5b5eb3	Download	ios	Phone	2025-02-19 18:14:59.866918
2	923aa092-a566-42ef-8e0a-77665f9e4bb0	019d87ce-756d-4a56-909f-dbaf0a5b5eb3	Install	ios	Phone	2025-02-19 23:14:59.866918
3	724935ea-012d-4a2c-85ff-15e0e790308e	98cedc91-5230-4b66-b7f2-edbffe8d8833	PageView	ios	Phone	2025-01-11 15:41:35.254317
4	051392b2-e239-4a4a-9dad-4730977d6776	3ca4b9f4-b223-4080-8de7-fce445b39789	PageView	ios	Phone	2025-01-18 22:12:25.750521
...	...	...	...	...	...	...
2503	8c3e66c6-b3ca-4169-bad5-c48fc7f0c852	7dec483a-0bd8-4b2c-ad10-2917d8b234c5	Install	android	Tablet	2025-02-15 22:06:05.192405
2504	3bc77700-4ae7-4e26-b8d7-8ce1eabbec52	c5f9cde9-5f20-428b-bd2c-b97497c84afa	PageView	ios	Tablet	2025-01-07 00:33:37.787415
2505	f4606926-9627-4d5d-b9cb-937c1085ba82	8ee7d879-8235-41a4-b77c-6634e15a447a	PageView	ios	Phone	2025-01-07 11:58:41.221669

	event_id	user_id	event_name	platform	device_type	timestamp
2506	be3978a4-90fd-4860-88e1-755cb5da99c6	8ee7d879-8235-41a4-b77c-6634e15a447a	Download	ios	Phone	2025-01-07 22:58:41.221669
2507	7ffbc348-5291-4cf7-a494-45e8e30b7207	8ee7d879-8235-41a4-b77c-6634e15a447a	Install	ios	Phone	2025-01-08 13:58:41.221669

2508 rows × 6 columns

Then, I added HardPaywall event after install just before the purchase. I think the number of users who see the app's HardPaywall is also important. Some apps conduct surveys when users first click on the app then Hard Paywall emerges just after the survey.

```
import pandas as pd
import random
from faker import Faker
from datetime import timedelta

fake = Faker()

# Possible values
event_names = ['PageView', 'Download', 'Install', 'HardPaywall', 'Purchase']
platforms = ['ios', 'android']
device_types = ['Phone', 'Tablet']

# The number of users
num_users = 1000  # Adjust as needed

# A list to store events
events = []

for _ in range(num_users):
    user_id = fake.uuid4()  # Unique user

    # 1: PageView (100% of users)
    pageview_timestamp = fake.date_time_this_year()
    platform = random.choice(platforms)
    device_type = random.choice(device_types)
```

```python
events.append({
    'event_id': fake.uuid4(),
    'user_id': user_id,
    'event_name': 'PageView',
    'platform': platform,
    'device_type': device_type,
    'timestamp': pageview_timestamp.strftime('%Y-%m-%d %H:%M:%S')
})

# 2: Download (80% chance)
if random.random() < 0.8:
    download_timestamp = pageview_timestamp + timedelta(hours=random.randint(1, 72))

    events.append({
        'event_id': fake.uuid4(),
        'user_id': user_id,
        'event_name': 'Download',
        'platform': platform,
        'device_type': device_type,
        'timestamp': download_timestamp.strftime('%Y-%m-%d %H:%M:%S')
    })

    # 3: Install (90% of those who downloaded)
    if random.random() < 0.9:
        install_timestamp = download_timestamp + timedelta(hours=random.randint(1, 72))

        events.append({
            'event_id': fake.uuid4(),
            'user_id': user_id,
            'event_name': 'Install',
            'platform': platform,
            'device_type': device_type,
            'timestamp': install_timestamp.strftime('%Y-%m-%d %H:%M:%S')
        })

        # 4: HardPaywall (90% of those who installed)
```

```
        if random.random() < 0.9:
            hardpaywall_timestamp = install_timestamp + timedelta(hours=random.randint(1, 72))

            events.append({
                'event_id': fake.uuid4(),
                'user_id': user_id,
                'event_name': 'HardPaywall',
                'platform': platform,
                'device_type': device_type,
                'timestamp': hardpaywall_timestamp.strftime('%Y-%m-%d %H:%M:%S')
            })

            # Step 5: Purchase (10% of those who saw HardPaywall)
            if random.random() < 0.1:
                purchase_timestamp = hardpaywall_timestamp + timedelta(hours=random.randint(1, 72))

                events.append({
                    'event_id': fake.uuid4(),
                    'user_id': user_id,
                    'event_name': 'Purchase',
                    'platform': platform,
                    'device_type': device_type,
                    'timestamp': purchase_timestamp.strftime('%Y-%m-%d %H:%M:%S')
                })


df_user_events = pd.DataFrame(events)

df_user_events.to_csv("user_events2.csv", index=False, encoding='utf-8', sep=',')

print(df_user_events.head())
```

```
                               event_id                               user_id  \
0  5773c7d0-2865-493b-bcc9-8306d6531f65  d84312b2-b0d6-4c69-a2e8-6f41a5464b84
1  6a4a7dab-d369-4753-9d6b-4123c232fe9b  d84312b2-b0d6-4c69-a2e8-6f41a5464b84
2  00c44c78-6fd9-48c4-ba57-5bb61432d165  d84312b2-b0d6-4c69-a2e8-6f41a5464b84
3  62acc1d1-fabb-4e53-b927-0f93d59ec886  d84312b2-b0d6-4c69-a2e8-6f41a5464b84
```

```
4  a4dd33e0-d4e2-47ee-a688-b146659c310d  cabd788f-b2c1-477a-9aed-e4e7d5a9edfc

  event_name platform device_type           timestamp
0    PageView      ios       Phone  2025-02-10 17:11:56
1    Download      ios       Phone  2025-02-13 04:11:56
2     Install      ios       Phone  2025-02-13 06:11:56
3  HardPaywall      ios       Phone  2025-02-15 19:11:56
4    PageView      ios      Tablet  2025-01-05 08:14:39
```

```
df_user_events["event_name"].value_counts()
```

```
event_name
PageView       1000
Download        807
Install         724
HardPaywall     644
Purchase         67
Name: count, dtype: int64
```

2- My SQL script (userevents1.sql):

```
        -- Creating Table
USE user_data; CREATE TABLE user_events (
   event_id VARCHAR(255) PRIMARY KEY,
   user_id VARCHAR(255),
   event_name ENUM('PageView', 'Download', 'Install', 'HardPaywall', 'Purchase'),
   platform ENUM('ios', 'android'),
   device_type VARCHAR(255),
   timestamp TIMESTAMP
);

ALTER TABLE user_events
```

```
MODIFY COLUMN event_name ENUM('PageView', 'Download', 'Install', 'HardPaywall',
'Purchase');


        -- Loading our dataset
USE user_data;
LOAD DATA LOCAL INFILE '/Users/computer/Desktop/Scripts/user_events2.csv'
INTO TABLE user_events
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;


SELECT COUNT(*) FROM user_events;
SELECT * FROM user_events LIMIT 10;
SHOW WARNINGS;
SHOW ERRORS;




USE user_data; SELECT COUNT(*) FROM user_events;

SELECT * FROM user_events;

select version();
```

```sql
-- Funnel Anaylsis
USE user_data;


DROP TEMPORARY TABLE IF EXISTS user_funnel;
DROP TEMPORARY TABLE IF EXISTS filtered_funnel;

-- Creating the 'user_funnel' Temporary Table
CREATE TEMPORARY TABLE user_funnel AS
SELECT
    user_id,
    platform,
    MIN(CASE WHEN event_name = 'PageView' THEN timestamp END) AS
pageview_time,
    MIN(CASE WHEN event_name = 'Download' THEN timestamp END) AS
download_time,
    MIN(CASE WHEN event_name = 'Install' THEN timestamp END) AS install_time
FROM user_events
WHERE event_name IN ('PageView', 'Download', 'Install')
GROUP BY user_id, platform;

-- Check if 'user_funnel' Table Created Successfully
SELECT * FROM user_funnel LIMIT 5;

-- Create the 'filtered_funnel' Temporary Table
CREATE TEMPORARY TABLE filtered_funnel AS
```

```sql
SELECT
    user_id,
    platform,
    pageview_time,
    download_time,
    install_time,

    -- Check if Download happened within 72 hours of PageView
    CASE
        WHEN download_time IS NOT NULL
        AND download_time <= pageview_time + INTERVAL 72 HOUR
        THEN 1 ELSE 0
    END AS converted_download,

    -- Check if Install happened within 72 hours of Download
    CASE
        WHEN install_time IS NOT NULL
        AND install_time <= download_time + INTERVAL 72 HOUR
        THEN 1 ELSE 0
    END AS converted_install
FROM user_funnel;

-- Check if 'filtered_funnel' Table Created Successfully
SELECT * FROM filtered_funnel LIMIT 5;


-- Performing the Funnel Analysis Query
SELECT
```

```
    platform,
    COUNT(DISTINCT user_id) AS total_users,
    COUNT(pageview_time) AS pageviews,
    COUNT(download_time) AS downloads,
    COUNT(install_time) AS installs,

    -- Valid conversions within 72 hours using conditional aggregation
    SUM(CASE WHEN converted_download = 1 THEN 1 ELSE 0 END) AS
valid_downloads,
    SUM(CASE WHEN converted_install = 1 THEN 1 ELSE 0 END) AS valid_installs,

    -- Conversion Rates
    ROUND(SUM(CASE WHEN converted_download = 1 THEN 1 ELSE 0 END) * 100.0
/ COUNT(pageview_time), 2) AS pageview_to_download_rate,
    ROUND(SUM(CASE WHEN converted_install = 1 THEN 1 ELSE 0 END) * 100.0 /
COUNT(download_time), 2) AS download_to_install_rate
FROM filtered_funnel
GROUP BY platform;


SHOW Tables;
SELECT * FROM user_events LIMIT 10;


SELECT
    platform,
    COUNT(DISTINCT user_id) AS total_users,
```

```sql
    COUNT(CASE WHEN event_name = 'PageView' THEN 1 END) AS pageviews,
    COUNT(CASE WHEN event_name = 'Download' THEN 1 END) AS downloads,
    COUNT(CASE WHEN event_name = 'Install' THEN 1 END) AS installs,
    ROUND(COUNT(CASE WHEN event_name = 'Download' THEN 1 END) * 100.0 /
COUNT(CASE WHEN event_name = 'PageView' THEN 1 END), 2) AS
pageview_to_download_rate,
    ROUND(COUNT(CASE WHEN event_name = 'Install' THEN 1 END) * 100.0 /
COUNT(CASE WHEN event_name = 'Download' THEN 1 END), 2) AS
download_to_install_rate
FROM user_events
WHERE event_name IN ('PageView', 'Download', 'Install')
GROUP BY platform;
```

3- UserFunnelAnalysisMetrics.csv

```
platform,total_users,pageviews,downloads,installs,pageview_to_download_rat
e,download_to_install_rate
ios,480,480,385,351,80.21,91.17
android,520,520,422,373,81.15,88.39
```

Part 2: Data Modeling Questions

1. Looking at the events data above, how would you model this data in a production environment? Consider aspects like:
Table structure, Partitioning strategy
What other tables might be needed? How would you handle data quality?
2. If we wanted to extend this analysis to include user attributes (like country, device type), what changes would you make to the data model?
3. What are potential issues with the current event tracking system that you can identify?

First of all, in a production environment, it is important to have well structured data. In order to manage large datasets, we need to partition the events table (either monthly or daily, depending on your strategy according to your specific project). Additionally, like what I did during this case, we have to index highly queried fields like timestamp to ensure maintenance.
As well as events table as primary, we may also use users and sessions tables. For me, user attributes data is also very important because I believe that demographics of users gives us

significant insights for building our model. Analyzing how different user segments engage with features can enable us to create personalized experiences and implement targeted optimizations. For the data quality, I regularly audit of the data pipeline and event tracking system to address issues proactively. Additionally, I implement detection alerts and develop automation processes to maintain stability and consistency in the system.

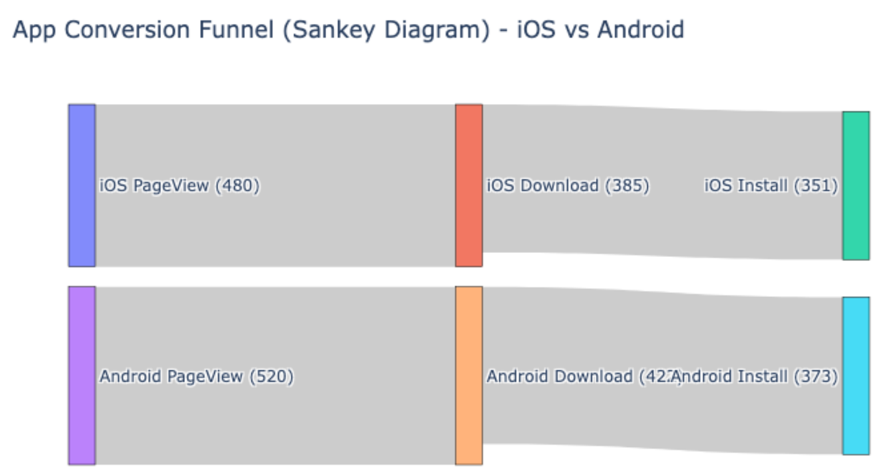
Part 3: Visualization

1. What tools would you recommend for visualizing this funnel data?
2. How would you design a dashboard to monitor these conversion rates over time?
3. What additional metrics or breakdowns would you include in the visualization?

Power BI and Tableau are good tools for visualization. It's easy to visualize my app data using these tools. But for me, using python libraries like; Matplotlib, Seaborn, Plotly, Dash, Streamlit gives you more free area and flexibility. So sometimes I use Python to visualize my data. I also have experience with Looker Studio. Looker Studio has many connectivity advantages and flexibility.

Apart from bar chart we can monitor conversion rates with Sankey diagram. It's better to see differences when using this diagram.

Here is an example:



Designing dashboards with Streamlit would be a good fit for my visualization. Since I have experience, it is very productive thanks to Python. By diversifying with smart chips, we can create many visualizations in it.

Many additional metrics can be implemented into my research such as adding HardPaywall and Purchase. Besides from them, we can add time between events to see how long it takes for a user to purchase our product. And of course, the churn rate should be added as well. Assume that we ask their ages to users after their first click to the app, if we get some insights about their demographics, then this would be a perfect breakdown opportunity for our model.