MOBILE APP ISSUE ANALYSIS USING SQL AND POWER BI

BY MANYA GUPTA

DATA

DATASET OF Users(users.csv) AND ISSUES (Issues.csv) OF MOBILE APPLICATION ARE **ARTIFICIALLY GENERATED (SYNTHETIC) .**

DATA PREPARATION IN SQL

1. FIRST CREATE DATABASE- **mobile_app_db .**
2. CREATE **TABLES users and issues** AND IMPORT THE DATA INTO THE TABLE BY USING **TABLE DATA IMPORT WIZARD .**
3. CLEAN THE DATA BY CHECKING MISSING VALUES (IF ANY) AND DUPLICATES.
4. USERS TABLE CONSISTS OF **1000 ROWS** AND **6 COLUMNS .**
   ISSUES TABLE CONSISTS OF **3000 ROWS** AND **8 COLUMNS .**

MOBILE APP ISSUE DATASET

THERE ARE TWO TABLES IN THE DATASET USERS AND ISSUES

1. USERS CONTAINS

COLUMN NAME	DESCRIPTION
USER_ID	UNIQUE ID FOR THE USER
NAME	FULL NAME OF THE USER
EMAIL	EMAIL ADDRESS
SIGNUP_DATE	DATE THE USER SIGNED UP
DEVICE_TYPE	TYPE OF DEVICE (ANDROID/IOS)
APP_VERSION	CURRENT VERSION OF THE APP

2. ISSUES CONTAINS

COLUMN NAME	DESCRIPTION
ISSUE_ID	UNIQUE ID FOR THE ISSUE
USER_ID	REFERENCES USERS.USER_ID
ISSUE_TYPE	TYPE OF ISSUE (CRASH, UI BUG, ETC.)
DESCRIPTION	DESCRIPTION OF THE ISSUE
REPORTED_DATE	TIMESTAMP WHEN THE ISSUE WAS REPORTED
STATUS	STATUS (OPEN, IN PROGRESS, RESOLVED)
PRIORITY	PRIORITY LEVEL (LOW, MEDIUM, HIGH, CRITICAL)
APP_VERSION	VERSION OF APP AT TIME OF ISSUE

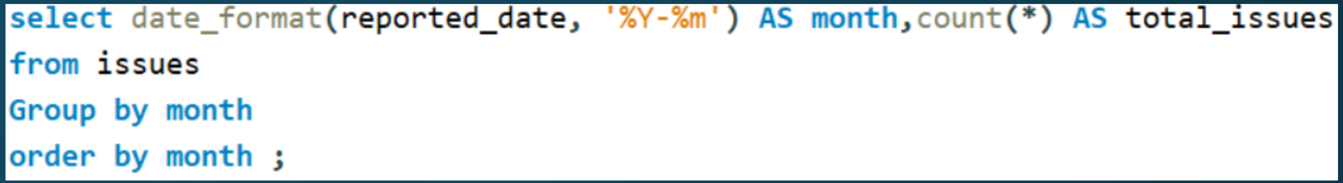
OBJECTIVE

TO ANALYZE THE TRENDS AND PATTERNS IN ISSUE REPORTS OVER TIME TO IDENTIFY THE MOST COMMON TYPES OF APP PROBLEMS, DETECT ANY SPIKES RELATED TO APP VERSIONS OR DEVICE TYPES, AND PROVIDE ACTIONABLE INSIGHTS TO IMPROVE APP STABILITY AND USER EXPERIENCE

RESEARCH QUESTIONS

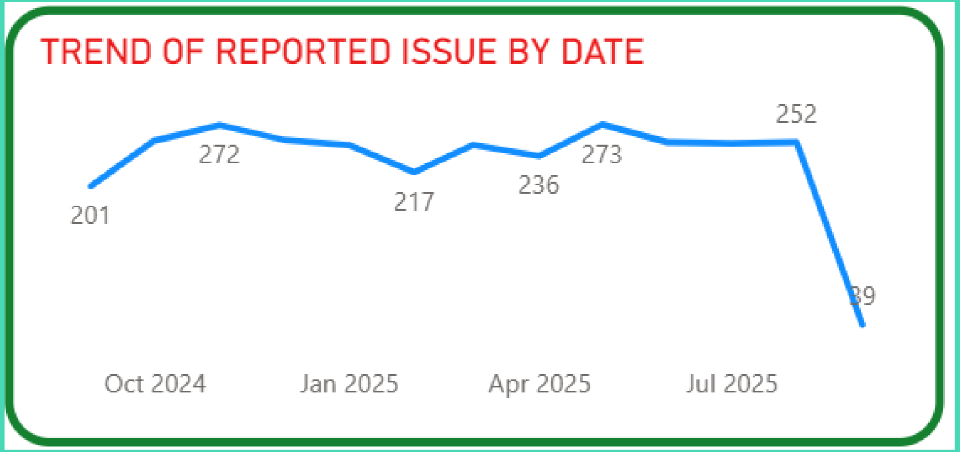
1. WHAT IS THE OVERALL TREND OF ISSUE OVER TIME ?
2. WHICH ISSUE TYPES (CRASH, UI BUG, PERFORMANCE, ETC.) ARE MOST FREQUENTLY REPORTED?
3. WHICH APP VERSIONS HAVE THE MOST REPORTED ISSUES?
4. ARE CERTAIN DEVICE TYPES MORE PRONE TO SPECIFIC ISSUE TYPES?
5. WHAT IS THE DISTRIBUTION OF ISSUE PRIORITY LEVELS (LOW, MEDIUM, HIGH, CRITICAL) OVER TIME?
6. HOW MANY ISSUES REMAIN UNRESOLVED (STATUS = OPEN OR IN PROGRESS) OVER TIME, AND IS THERE A BACKLOG GROWING?

1. **WHAT** IS THE OVERALL TREND OF ISSUE OVER TIME ?



```
select date_format(reported_date, '%Y-%m') AS month,count(*) AS total_issues
from issues
Group by month
order by month ;
```

THE TOTAL NUMBER OF ISSUES REPORTED **VARIES MONTH TO MONTH** — GOING UP AND DOWN, NOT A STEADY TREND



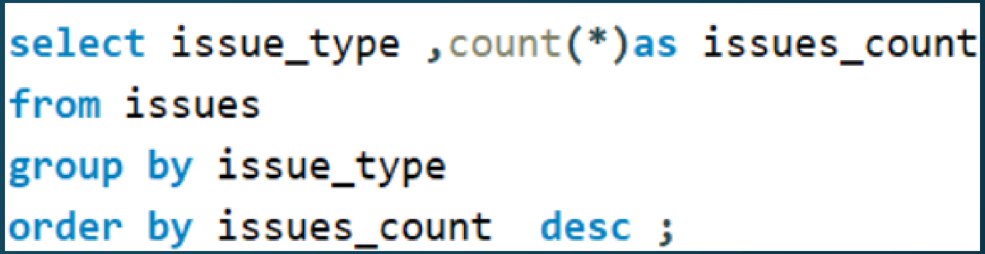
THE NUMBERS **WENT UP FROM 201 TO A HIGH OF 272,** THEN **WENT UP AND DOWN** AGAIN, REACHING ANOTHER **PEAK** OF **273** IN **MAY 2025.** AFTER STAYING STEADY FOR **TWO MONTHS,** THEY DROPPED TO **39** IN THE **LAST MONTH.**

THIS SHOWS THAT THE NUMBER OF REPORTED ISSUES CAN RISE QUICKLY BUT ALSO DROP JUST AS FAST.

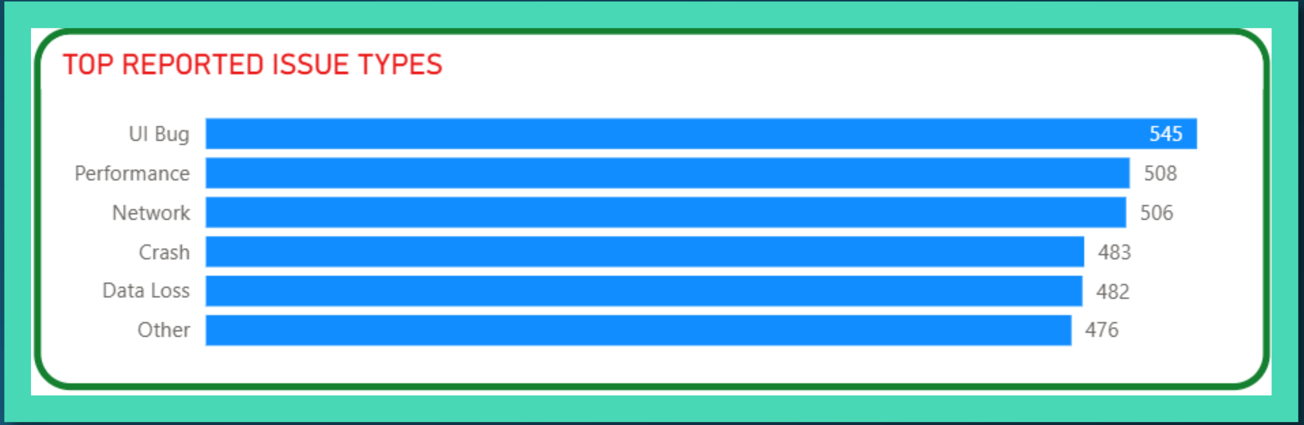
2. WHICH ISSUE TYPES (CRASH , UI BUG , PERFORMANCE ETC.) ARE MOST FREQUENTLY REPORTED

UI BUGS WERE REPORTED THE MOST, WITH **545 ISSUES**.

THE NUMBER OF REPORTS FOR EACH TYPE IS SIMILAR, BUT **UI BUGS ARE SLIGHTLY HIGHER THAN OTHERS.**



```
select issue_type ,count(*)as issues_count
from issues
group by issue_type
order by issues_count  desc ;
```

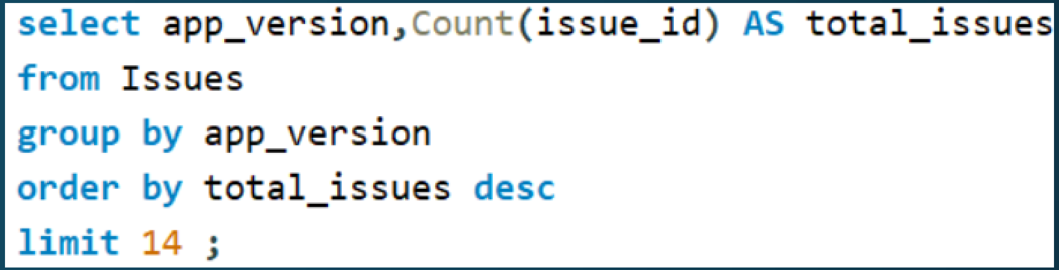


3. WHICH APP VERSION HAVE THE MOST REPORTED ISSUE ?

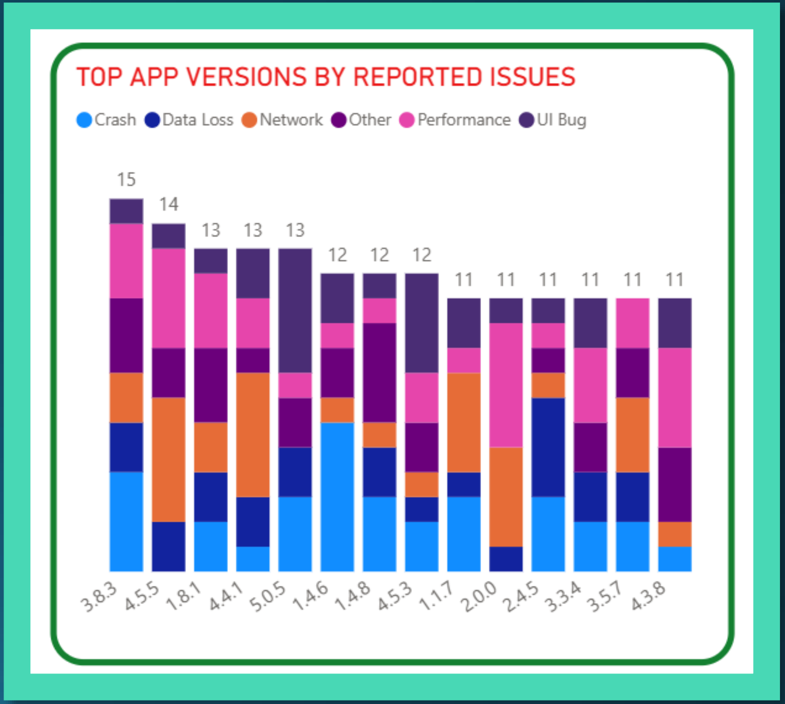
THE APP VERSION WITH THE MOST REPORTED ISSUES IS **3.8.3**, WITH **15 ISSUES**.

VERSIONS **4.5.5 (14 ISSUES)** AND **5.0.5 (13 ISSUES)** ALSO HAD HIGH ISSUE COUNTS.

SEVERAL OLDER VERSIONS (E.G., **1.8.1**, **1.4.6**) STILL SHOW **FREQUENT ISSUES**, SUGGESTING USERS ARE NOT UPDATING REGULARLY OR OLD BUGS PERSIST.



```
select app_version,Count(issue_id) AS total_issues
from Issues
group by app_version
order by total_issues desc
limit 14 ;
```



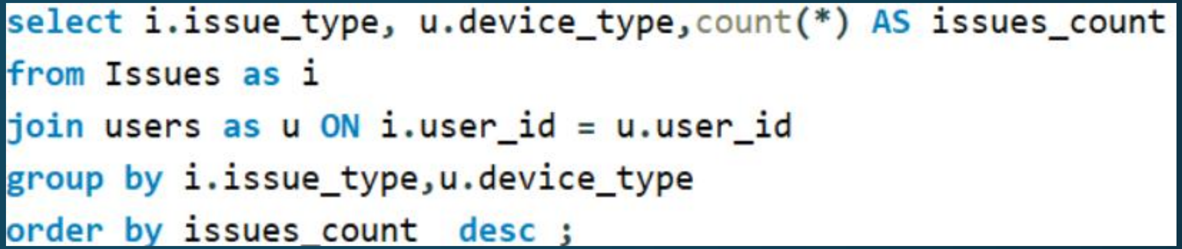
4. ARE CERTAIN DEVICE TYPES MORE PRONE TO SPECIFIC ISSUE TYPES?

IOS USERS REPORTED SLIGHTLY MORE ISSUES THAN **ANDROID** USERS IN MOST CATEGORIES.

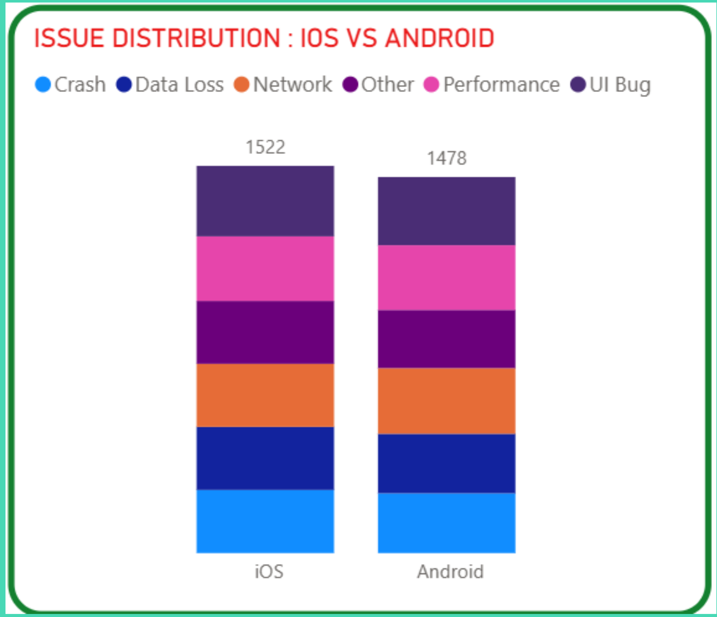
DIFFERENCES WERE IN:
UI BUGS: IOS (277) VS ANDROID (268)
DATA LOSS: IOS (248) VS ANDROID (234)
OTHER ISSUES: IOS (247) VS ANDROID (229)

PERFORMANCE ISSUES WERE **EQUAL** ON BOTH PLATFORMS (**254 EACH**).

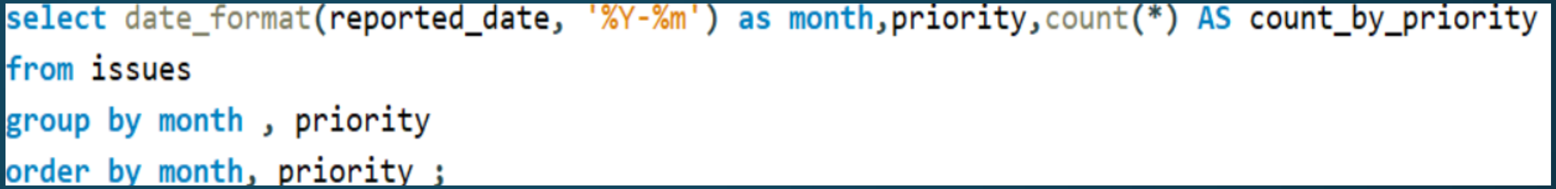
DIFFERENCES ARE **NOT EXTREME**, BUT IOS SHOWS **CONSISTENTLY HIGHER COUNTS**.



```
select i.issue_type, u.device_type,count(*) AS issues_count
from Issues as i
join users as u ON i.user_id = u.user_id
group by i.issue_type,u.device_type
order by issues_count  desc ;
```



5. WHAT IS THE DISTRIBUTION OF ISSUE PRIORITY LEVELS (LOW, MEDIUM, HIGH, CRITICAL) OVER TIME?

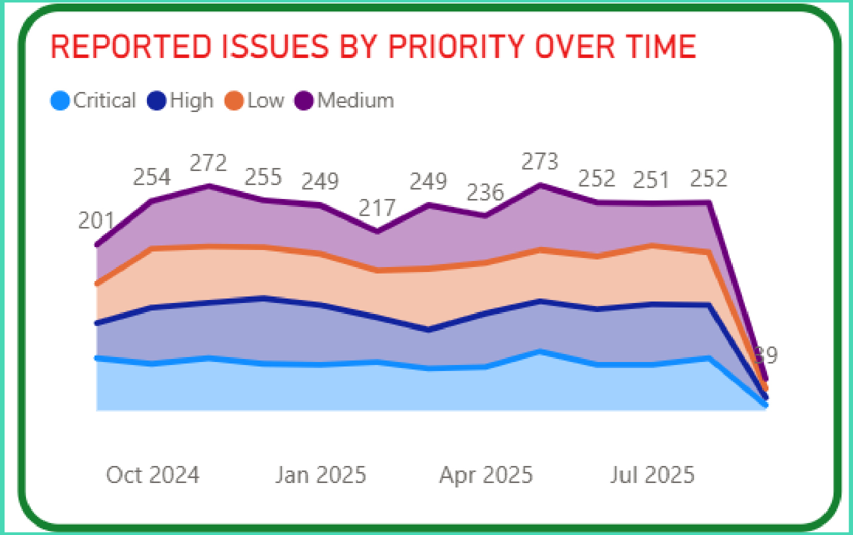


```
select date_format(reported_date, '%Y-%m') as month,priority,count(*) AS count_by_priority
from issues
group by month , priority
order by month, priority ;
```

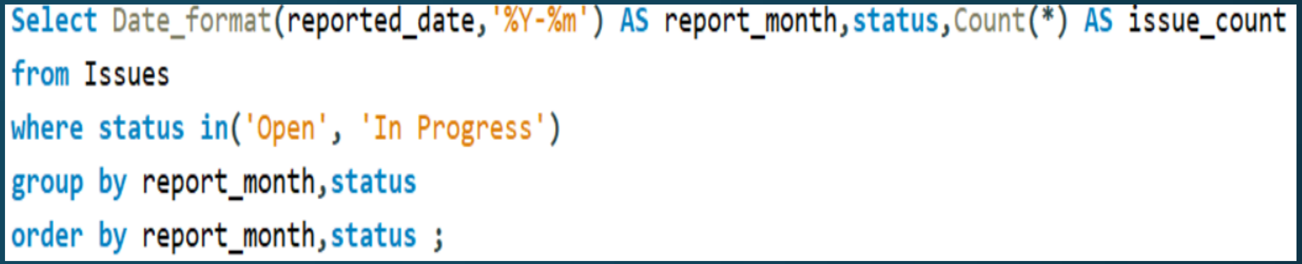
THE **NUMBER OF ISSUES WENT UP AND DOWN OVER TIME**, NOT A STEADY TREND.

MEDIUM AND HIGH PRIORITY ISSUES MADE UP MOST OF THE REPORTED PROBLEMS.

ALL ISSUES DROPPED SHARPLY AT THE END, WHICH COULD BE DUE TO ACTUAL IMPROVEMENT



6. HOW MANY ISSUES REMAIN UNRESOLVED (STATUS = OPEN OR IN PROGRESS) OVER TIME, AND IS THERE A BACKLOG GROWING?

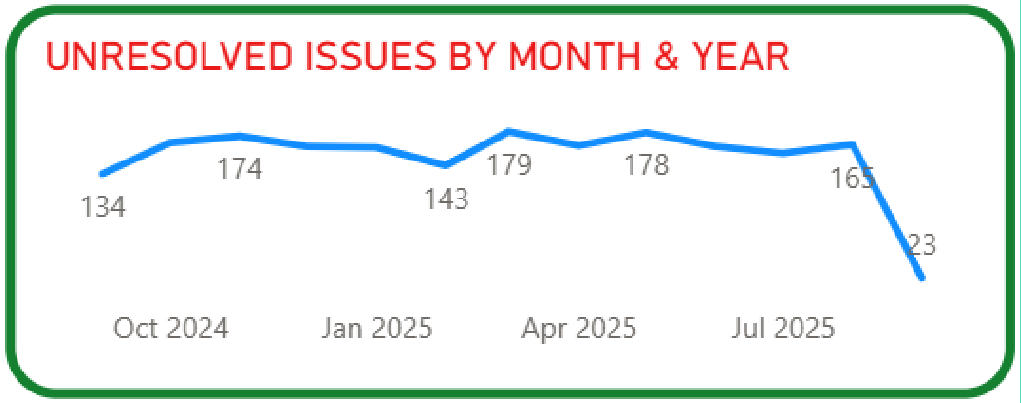


```
Select Date_format(reported_date,'%Y-%m') AS report_month,status,Count(*) AS issue_count
from Issues
where status in('Open', 'In Progress')
group by report_month,status
order by report_month,status ;
```

UNRESOLVED ISSUES WENT **UP AND DOWN** OVER TIME — NOT A STEADY TREND.

THE HIGHEST POINT WAS AROUND **MARCH 2025**, THEN DROPPED SHARPLY.

THE BIG DROP AT THE END MAY BE **REAL PROGRESS**



MOBILE APP ISSUE ANALYTICS DASHBOARD

06-09-2024

06-09-2025

TOTAL ISSUES REPORTED	CRITICAL ISSUES	UNRESOLVED ISSUES	BACKLOG %	USERS REPORTING ISSUES
3000	716	1971	66%	951

issue_ty...

All

Android

iOS

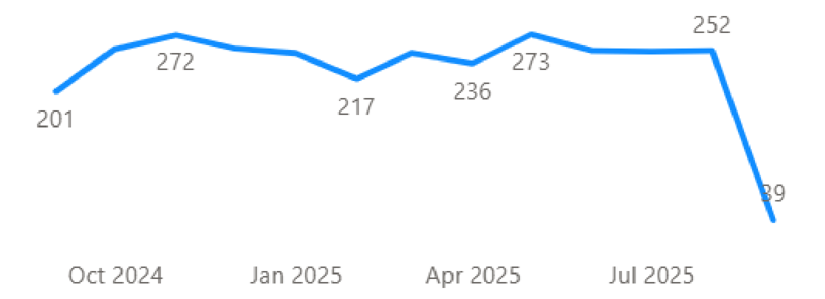
Critical

High

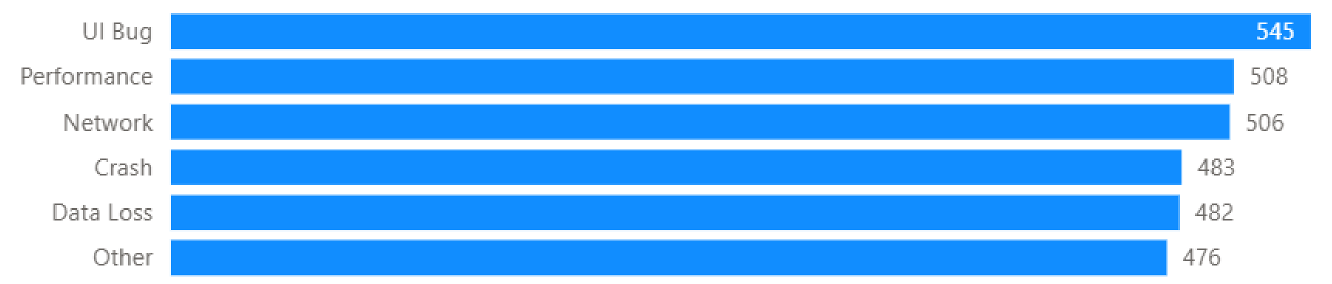
Low

Medium

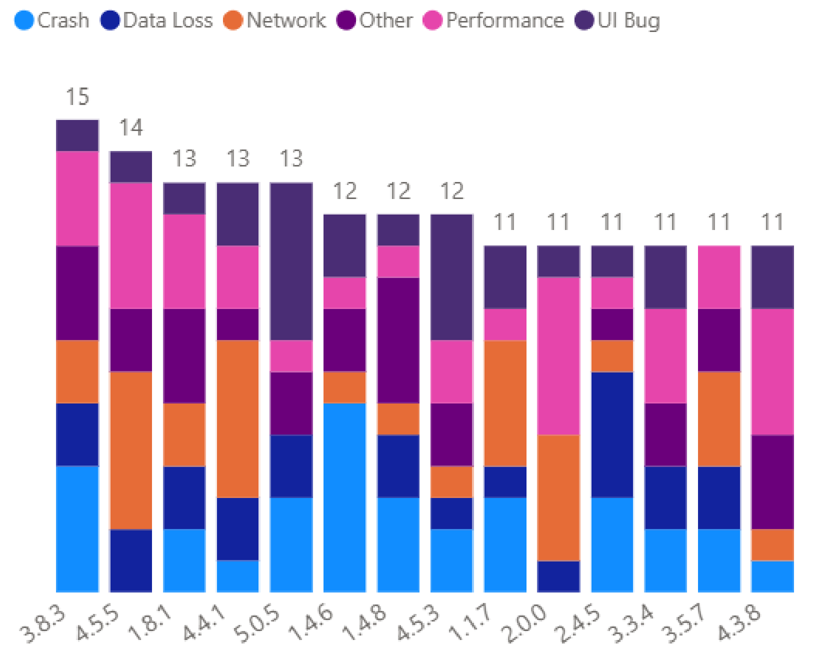
TREND OF REPORTED ISSUE BY DATE



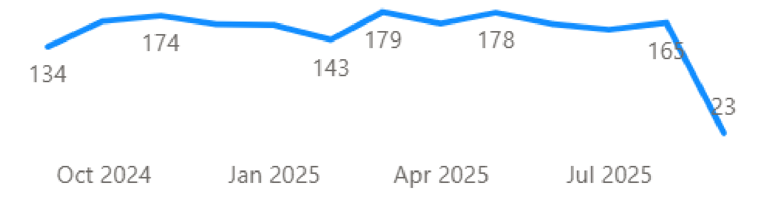
TOP REPORTED ISSUE TYPES



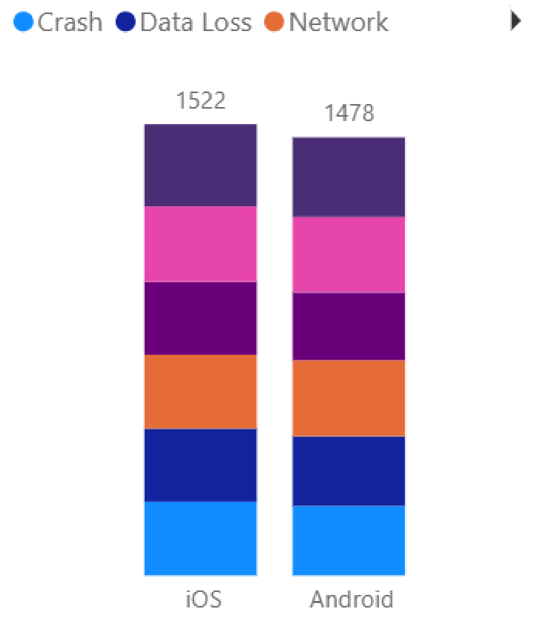
TOP APP VERSIONS BY REPORTED ISSUES



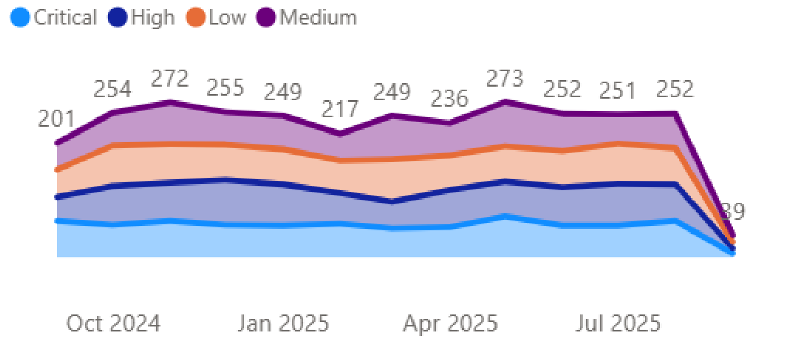
UNRESOLVED ISSUES BY MONTH & YEAR



ISSUE DISTRIBUTION : IOS VS ANDROID

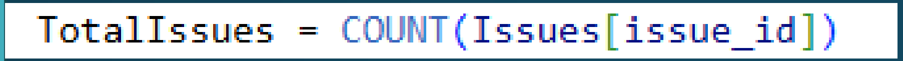


REPORTED ISSUES BY PRIORITY OVER TIME



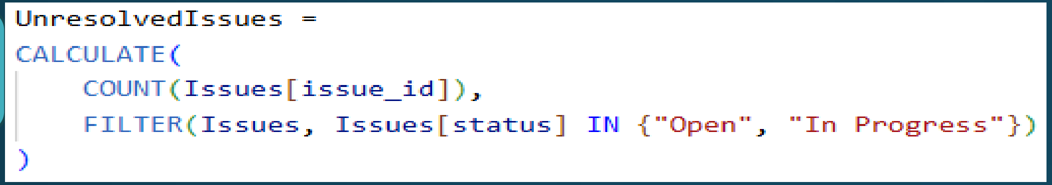
KPI

TOTAL ISSUES REPORTED



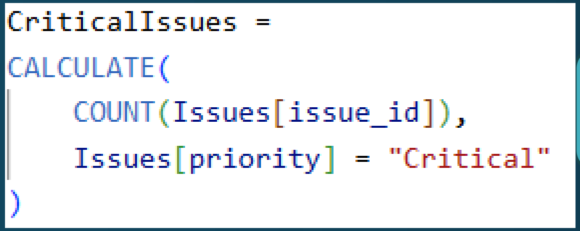
```
TotalIssues = COUNT(Issues[issue_id])
```

UNRESOLVED ISSUES



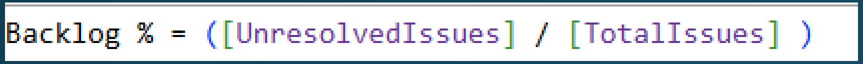
```
UnresolvedIssues =
CALCULATE(
    COUNT(Issues[issue_id]),
    FILTER(Issues, Issues[status] IN {"Open", "In Progress"})
)
```

CRITICAL ISSUES



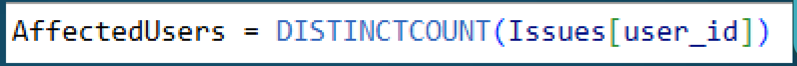
```
CriticalIssues =
CALCULATE(
    COUNT(Issues[issue_id]),
    Issues[priority] = "Critical"
)
```

BACKLOG %



```
Backlog % = ([UnresolvedIssues] / [TotalIssues] )
```

USERS REPORTING ISSUE



```
AffectedUsers = DISTINCTCOUNT(Issues[user_id])
```

SLICERS

ISSUE TYPES

DEVICE

PRIORITY

REPORTED DATE

RECOMMENDATIONS

KEEP CHECKING THE NUMBER OF ISSUES TO SEE WHY THEY GO UP. MAKE SURE THE RECENT DROP IS REAL AND NOT A MISTAKE.

FIX UI BUGS FIRST BECAUSE THEY ARE REPORTED THE MOST. ALSO, CHECK PERFORMANCE AND NETWORK ISSUES, AS THEY HAPPEN OFTEN AND IMPROVING THEM CAN MAKE USERS MORE SATISFIED.

LOOK INTO VERSIONS 3.8.3, 4.5.5, AND 5.0.5 FOR POSSIBLE BUGS. ASK USERS TO UPDATE IF OLDER VERSIONS HAVE ISSUES, AND THINK ABOUT STOPPING SUPPORT FOR VERSIONS THAT ARE RARELY USED AND STILL HAVE PROBLEMS.

RECOMMENDATIONS

TEST THE APP MORE ON IOS DEVICES, ESPECIALLY FOR UI AND DATA PROBLEMS. MAKE SURE BOTH ANDROID AND IOS VERSIONS ARE PROPERLY CHECKED BEFORE RELEASING UPDATES.

FIX HIGH AND CRITICAL ISSUES QUICKLY TO REDUCE RISK. KEEP AN EYE ON TRENDS, ESPECIALLY CRITICAL ONES, AND CHECK IF THE RECENT DROP IS REAL OR JUST A REPORTING ISSUE.

KEEP UP THE GOOD WORK TO KEEP UNRESOLVED ISSUES LOW. WATCH THE BACKLOG CLOSELY, AND IMPROVE WORKFLOWS OR ADD HELP IF IT STARTS GROWING AGAIN.