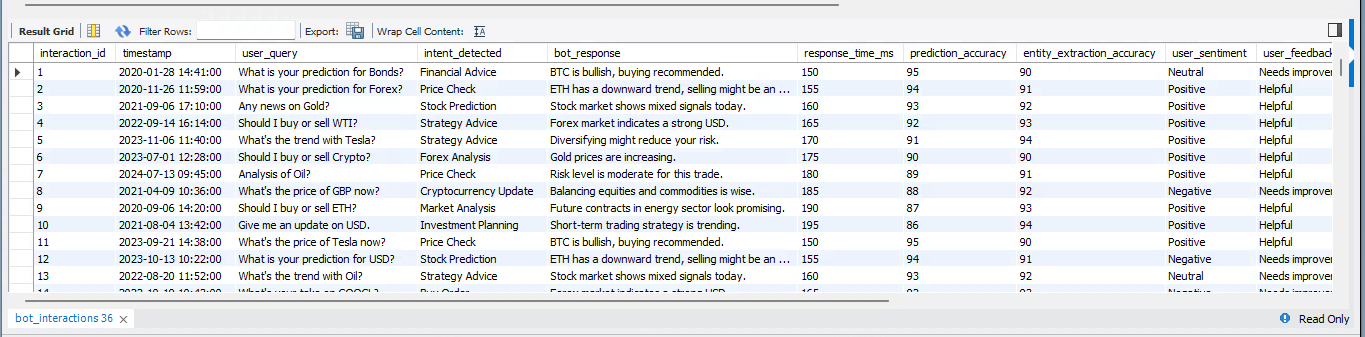
**INSERT QUERY OUTPUT**



1. **SQL Queries (For Custom Reports)**
2. **Top 5 Slowest Intents (Average Response Time)**

SELECT

intent\_detected,

ROUND(AVG(response\_time\_ms), 2) AS avg\_response\_time

FROM bot\_interactions

GROUP BY intent\_detected

ORDER BY avg\_response\_time DESC

LIMIT 5;

**O/P:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Feedback Score by Region and User Type**

SELECT

region,

user\_type,

user\_feedback,

COUNT(\*) AS total\_feedbacks

FROM bot\_interactions

GROUP BY region, user\_type, user\_feedback

ORDER BY total\_feedbacks DESC;

**O/P:**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Intent Accuracy Over Time (Monthly Trend)**

SELECT

DATE\_FORMAT(timestamp, '%Y-%m') AS month,

intent\_detected,

ROUND(AVG(prediction\_accuracy), 2) AS avg\_prediction\_accuracy

FROM bot\_interactions

GROUP BY month, intent\_detected

ORDER BY month;

**O/P:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Failed Conversations (Conversation Success = FALSE)**

SELECT

intent\_detected,

COUNT(\*) AS failed\_conversations

FROM bot\_interactions

WHERE conversation\_success = “Failed”

GROUP BY intent\_detected

ORDER BY failed\_conversations DESC;

**O/P:**

**A screenshot of a table

AI-generated content may be incorrect.**

1. **Cursor + Stored Procedure: Analyze Feedback Per Intent**

**Task:**

For each `intent\_detected`, calculate:

\* Total interactions

\* Helpful feedback count

\* Needs improvement count

DROP PROCEDURE IF EXISTS AnalyzeFeedbackPerIntent;

DELIMITER $$

CREATE PROCEDURE AnalyzeFeedbackPerIntent()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE intent\_name VARCHAR(100);

DECLARE intent\_cursor CURSOR FOR

SELECT DISTINCT intent\_detected FROM bot\_interactions;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

CREATE TEMPORARY TABLE IF NOT EXISTS intent\_feedback\_summary (

intent VARCHAR(100),

total INT,

helpful INT,

needs\_improvement INT

);

OPEN intent\_cursor;

read\_loop: LOOP

FETCH intent\_cursor INTO intent\_name;

IF done THEN

LEAVE read\_loop;

END IF;

INSERT INTO intent\_feedback\_summary

SELECT

intent\_name,

COUNT(\*) AS total,

SUM(CASE WHEN user\_feedback = 'Helpful' THEN 1 ELSE 0 END) AS helpful,

SUM(CASE WHEN user\_feedback = 'Needs Improvement' THEN 1 ELSE 0 END) AS needs\_improvement

FROM bot\_interactions

WHERE intent\_detected = intent\_name;

END LOOP;

CLOSE intent\_cursor;

SELECT \* FROM intent\_feedback\_summary;

END$$

DELIMITER ;

CALL AnalyzeFeedbackPerIntent();

**O/P:**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Function: Response Quality Score (0–1 scale)**

DROP FUNCTION IF EXISTS GetResponseQualityScore;

DELIMITER $$

CREATE FUNCTION GetResponseQualityScore(

acc FLOAT,

entity\_acc FLOAT,

response\_time INT

) RETURNS FLOAT

DETERMINISTIC

BEGIN

DECLARE score FLOAT;

SET score = ((acc + entity\_acc) / 2) \* (1 - (response\_time / 2000.0));

RETURN GREATEST(0, LEAST(1, score));

END$$

DELIMITER ;

SELECT

interaction\_id,

GetResponseQualityScore(prediction\_accuracy, entity\_extraction\_accuracy, response\_time\_ms) AS quality\_score

FROM bot\_interactions

ORDER BY quality\_score DESC

LIMIT 10;

**O/P:**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Report Stored Procedure: Premium Users Only**

DROP PROCEDURE IF EXISTS PremiumUserReport;

DELIMITER $$

CREATE PROCEDURE PremiumUserReport()

BEGIN

SELECT

user\_id,

user\_type,

intent\_detected,

COUNT(\*) AS total\_interactions,

AVG(response\_time\_ms) AS avg\_response\_time,

SUM(CASE WHEN conversation\_success = “Successful” THEN 1 ELSE 0 END) AS success\_count

FROM bot\_interactions

WHERE is\_premium = TRUE

GROUP BY user\_id, intent\_detected;

END$$

DELIMITER ;

CALL PremiumUserReport();

**O/P:**

**A screenshot of a data sheet

AI-generated content may be incorrect.**

1. **ADVANCED SQL QUERIES FOR BOT ANALYSIS**
2. **Overall Bot Performance Summary**

SELECT

COUNT(\*) AS total\_interactions,

ROUND(AVG(response\_time\_ms), 2) AS avg\_response\_time,

ROUND(AVG(prediction\_accuracy), 2) AS avg\_prediction\_accuracy,

ROUND(AVG(entity\_extraction\_accuracy), 2) AS avg\_entity\_accuracy,

SUM(CASE WHEN conversation\_success = “Successful” THEN 1 ELSE 0 END) AS successful\_conversations,

SUM(CASE WHEN conversation\_success = “Failed” THEN 1 ELSE 0 END) AS failed\_conversations

FROM bot\_interactions;

**O/P:**



1. **Feedback Breakdown by Device Type**

SELECT

device\_type,

user\_feedback,

COUNT(\*) AS feedback\_count

FROM bot\_interactions

GROUP BY device\_type, user\_feedback

ORDER BY device\_type, feedback\_count DESC;

**O/P:**

A screenshot of a survey

AI-generated content may be incorrect.

1. **Peak Interaction Hours**

SELECT

HOUR(timestamp) AS hour,

COUNT(\*) AS interaction\_count

FROM bot\_interactions

GROUP BY hour

ORDER BY interaction\_count DESC;

**O/P:**

A screenshot of a survey

AI-generated content may be incorrect.

1. **Average Accuracy by Region**

SELECT

region,

ROUND(AVG(prediction\_accuracy), 2) AS avg\_prediction\_accuracy,

ROUND(AVG(entity\_extraction\_accuracy), 2) AS avg\_entity\_accuracy

FROM bot\_interactions

GROUP BY region

ORDER BY avg\_prediction\_accuracy DESC;

**O/P:**

A screenshot of a graph

AI-generated content may be incorrect.

1. **Intents with Highest Negative Feedback**

SELECT

intent\_detected,

COUNT(\*) AS total,

SUM(CASE WHEN user\_feedback = 'Needs Improvement' THEN 1 ELSE 0 END) AS negative\_feedbacks,

ROUND(100.0 \* SUM(CASE WHEN user\_feedback = 'Needs Improvement' THEN 1 ELSE 0 END) / COUNT(\*), 2) AS negative\_feedback\_pct

FROM bot\_interactions

GROUP BY intent\_detected

HAVING COUNT(\*) > 10

ORDER BY negative\_feedback\_pct DESC;

**O/P:**

A screenshot of a data

AI-generated content may be incorrect.

1. **Premium vs Non-Premium User Feedback Comparison**

SELECT

is\_premium,

user\_feedback,

COUNT(\*) AS feedback\_count

FROM bot\_interactions

GROUP BY is\_premium, user\_feedback

ORDER BY is\_premium, feedback\_count DESC;

**O/P:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Sentiment Analysis Distribution**

SELECT

user\_sentiment,

COUNT(\*) AS sentiment\_count

FROM bot\_interactions

GROUP BY user\_sentiment

ORDER BY sentiment\_count DESC;

**O/P:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Average Response Time per Feedback Type**

SELECT

user\_feedback,

ROUND(AVG(response\_time\_ms), 2) AS avg\_response\_time

FROM bot\_interactions

GROUP BY user\_feedback;

**O/P:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Top 10 Users with Most Conversations**

SELECT

user\_id,

COUNT(\*) AS conversation\_count

FROM bot\_interactions

GROUP BY user\_id

ORDER BY conversation\_count DESC

LIMIT 10;

**O/P:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Device Type vs. Conversation Success Rate**

SELECT

device\_type,

COUNT(\*) AS total\_interactions,

SUM(CASE WHEN conversation\_success = “Successful” THEN 1 ELSE 0 END) AS successful,

ROUND(100.0 \* SUM(CASE WHEN conversation\_success = “Successful” THEN 1 ELSE 0 END) / COUNT(\*), 2) AS success\_rate

FROM bot\_interactions

GROUP BY device\_type

ORDER BY success\_rate DESC;

**O/P:**

A screenshot of a computer

AI-generated content may be incorrect.

1. **Account Age Influence on Feedback**

SELECT

CASE

WHEN account\_age\_months < 6 THEN '0-6 Months'

WHEN account\_age\_months BETWEEN 6 AND 12 THEN '6-12 Months'

WHEN account\_age\_months BETWEEN 13 AND 24 THEN '1-2 Years'

ELSE '2+ Years'

END AS age\_group,

user\_feedback,

COUNT(\*) AS feedback\_count

FROM bot\_interactions

GROUP BY age\_group, user\_feedback

ORDER BY age\_group;

**O/P:**

A screenshot of a survey

AI-generated content may be incorrect.