Student Homepage

SQL Query Documentation

Complete Database Mapping & Implementation Guide

Introduction

This document provides comprehensive documentation for all SQL queries used in the Student Homepage feature. Each query includes:

• Database tables and columns used

• Complete SQL query with parameters

• Output description in simple terms

• Important notes and limitations (caveats)

Section 1: Next Lesson Information

This section displays information about the student's upcoming lesson.

Index	1.0
Section & Heading	Top Section – Next Lesson
Homepage Component	Next Lesson Card (Day, Date, Time)
Database Tables	classes.meeting_start
SQL Query	`SELECT c.id AS class_id, DATE_FORMAT(c.meeting_start, '%W') AS day_name, DATE_FORMAT(c.meeting_start, '%Y-%m-%d') AS lesson_date, DATE_FORMAT(c.meeting_start, '%H:%i') AS lesson_time, c.meeting_start AS full_datetime FROM classes c WHERE c.student_id = {{student_id}} AND c.status = 'pending' AND c.meeting_start >= NOW() ORDER BY c.meeting_start ASC LIMIT 1;`
Output	Shows the next upcoming lesson with day (e.g., 'Monday'), date (e.g., '2026-01-30'), and time (e.g., '14:30'). This creates the main lesson card that students see first.
Caveat	Uses {{student_id}} parameter that must be provided by the API. Only shows 'pending' lessons that haven't started yet.

Index	1.1
Section & Heading	Teacher Information
Homepage Component	Teacher Name + Avatar

Database Tables	users.full_name, users.avatar
SQL Query	`SELECT u.full_name AS teacher_name, u.avatar AS teacher_avatar FROM classes c INNER JOIN users u ON c.teacher_id = u.id WHERE c.student_id = {{student_id}} AND c.status = 'pending' AND c.meeting_start >= NOW() ORDER BY c.meeting_start ASC LIMIT 1;`
Output	Displays the teacher's name and profile picture for the next lesson. This helps students know who they'll be learning with.
Caveat	Requires {{student_id}} parameter. Avatar field contains the image URL or path.

Index	1.2
Section & Heading	Join Lesson CTA
Homepage Component	Join Lesson CTA (Status Logic)
Database Tables	classes.join_url, classes.status
SQL Query	`SELECT c.id AS class_id, c.join_url, c.status, c.meeting_start, CASE WHEN c.meeting_start <= NOW() + INTERVAL 5 MINUTE AND c.meeting_start >= NOW() THEN 'enabled' ELSE 'disabled' END AS join_button_status FROM classes c WHERE c.student_id = {{student_id}} AND c.status = 'pending' AND c.meeting_start >= NOW() ORDER BY c.meeting_start ASC LIMIT 1;`

Output	Shows the Join Lesson button and determines if it should be clickable. Button is enabled only 5 minutes before lesson starts. Includes the Zoom/meeting link URL.
Caveat	Join button logic: enabled 5 minutes before lesson starts, disabled otherwise. Time window can be adjusted by changing the INTERVAL value.

Index	1.3
Section & Heading	Secondary Actions
Homepage Component	Secondary Actions (View schedule, cancel)
Database Tables	classes.id, classes.status
SQL Query	`SELECT c.id AS class_id, c.status, CASE WHEN c.status = 'pending' AND TIMESTAMPDIFF(HOUR, NOW(), c.meeting_start) >= 4 THEN 1 ELSE 0 END AS can_cancel FROM classes c WHERE c.student_id = {{student_id}} AND c.status = 'pending' AND c.meeting_start > NOW() ORDER BY c.meeting_start ASC LIMIT 1;`
Output	Determines if the student can cancel their lesson. Returns 1 if cancellation is allowed, 0 if not. Used to show/hide the cancel button.
Caveat	Cancellation is only allowed 4 hours before the lesson. This 24-hour policy is hardcoded and can be changed by adjusting the TIMESTAMPDIFF threshold.

Index	1.4
Section & Heading	Classes Usage
Homepage Component	Classes Usage (Classes used vs available)
Database Tables	user_subscription_details.left_lessons, user_subscription_details.weekly_lesson
SQL Query	`SELECT left_lessons AS classes_remaining, weekly_lesson AS total_weekly_lessons, (weekly_lesson - left_lessons) AS classes_used FROM user_subscription_details WHERE user_id = {{current_user_id}} AND status = 'active' ORDER BY created_at DESC LIMIT 1;`
Output	Shows how many classes the student has completed versus their total weekly allowance. For example: 'Used 3 of 5 weekly classes'. Helps students track their subscription usage.
Caveat	Requires {{current_user_id}} parameter. Only shows active subscriptions. If a student has multiple subscriptions, shows the most recent one.

Index	1.5
Section & Heading	No Lesson State
Homepage Component	No Lesson State (Book lesson CTA)

Database Tables	classes.status
SQL Query	`SELECT {{student_id}} AS student_id, CASE WHEN EXISTS (SELECT 1 FROM classes WHERE student_id = {{student_id}} AND status = 'pending' AND meeting_start > NOW()) THEN 1 ELSE 0 END AS has_pending_lesson, CASE WHEN NOT EXISTS (SELECT 1 FROM classes WHERE student_id = {{student_id}} AND status = 'pending' AND meeting_start > NOW()) THEN 1 ELSE 0 END AS show_book_cta;`
Output	Checks if student has any upcoming lessons. If no lessons scheduled, shows a 'Book a Lesson' button. Returns 1 to show button, 0 to hide it.
Caveat	Logic-based query that determines UI state. No lesson data needed, just checks if any pending lessons exist.

Section 2: Learning Progress Snapshot

This section shows the student's current English level and learning progress.

Index	2.0
Section & Heading	Learning Progress Snapshot
Homepage Component	Current Level (CEFR scale anchor)

Database Tables	student_progress.current_level, level_assessments.detected_level
SQL Query	`SELECT sp.student_id, sp.current_level, COALESCE(la.detected_level, sp.current_level) AS detected_level FROM student_progress sp LEFT JOIN (SELECT la2.student_id, la2.detected_level FROM level_assessments la2 INNER JOIN (SELECT student_id, MAX(assessed_at) as max_assessed FROM level_assessments GROUP BY student_id) latest ON la2.student_id = latest.student_id AND la2.assessed_at = latest.max_assessed) la ON sp.student_id = la.student_id WHERE sp.student_id = {{student_id}};`
Output	Shows the student's current CEFR level (A1, A2, B1, B2, C1, C2). Uses the most recent assessment if available, otherwise uses the stored current level.
Caveat	CEFR = Common European Framework of Reference. Levels range from A1 (beginner) to C2 (proficient). Uses most recent assessment date.

Index	2.1
Section & Heading	Level Meaning
Homepage Component	Level Meaning (One-line explanation)
Database Tables	Not possible from existing database (Static mapping)
SQL Query	`N/A - Static text mapping in frontend`

Output	Provides a simple explanation of what each CEFR level means. For example: 'A2 - Can communicate in simple routine tasks'. This is static text, not database-driven.
Caveat	Not a database query. Level descriptions are hardcoded in the application based on CEFR level (A1-C2).

Index	2.2
Section & Heading	Current Learning Goal
Homepage Component	Current Learning Goal (Personalized goal)
Database Tables	user_goals.goal_name, classes.student_goal
SQL Query	`SELECT u.id AS student_id, COALESCE(ug.goal_name, c.student_goal, 'Improve English fluency') AS goal_name FROM users u LEFT JOIN user_goals ug ON u.id = ug.user_id LEFT JOIN (SELECT student_id, student_goal FROM classes WHERE student_goal IS NOT NULL ORDER BY meeting_start DESC LIMIT 1) c ON u.id = c.student_id WHERE u.id = {{student_id}};`
Output	Shows what the student is currently working towards. For example: 'Improve business English communication' or 'Prepare for IELTS exam'. Uses student's goal if set, otherwise uses default.
Caveat	Priority: 1) User's saved goal, 2) Most recent class-specific goal, 3) Default 'Improve English fluency'. Requires {{student_id}} parameter.

Index	2.3
Section & Heading	Progress Visualization
Homepage Component	Progress Visualization (3-5 Checkpoints)
Database Tables	class_summaries.topics_detected, topics_taught.topic_name
SQL Query	`SELECT tt.id AS topic_id, tt.topic_name, CASE WHEN tt.verified_by_teacher = 1 THEN 1 ELSE 0 END AS is_complete FROM topics_taught tt INNER JOIN class_summaries cs ON tt.summary_id = cs.id INNER JOIN classes c ON cs.class_id = c.id WHERE c.student_id = {{student_id}} ORDER BY tt.created_at DESC LIMIT 5;`
Output	Shows the last 5 topics the student has learned. Creates a visual progress timeline with checkpoints. For example: 'Past Simple Tense ✓', 'Business Vocabulary ✓', 'Present Perfect'.
Caveat	Limited to 5 most recent topics. is_complete shows if teacher verified the topic (1 = complete, 0 = in progress).

Index	2.4
Section & Heading	Upcoming Learning Focus
Homepage Component	Upcoming Learning Focus (Planned direction)

Database Tables	class_summaries.areas_for_improvement
SQL Query	`SELECT cs.class_id, cs.areas_for_improvement FROM class_summaries cs INNER JOIN classes c ON cs.class_id = c.id WHERE c.student_id = {{student_id}} AND cs.areas_for_improvement IS NOT NULL ORDER BY c.meeting_start DESC LIMIT 1;`
Output	Shows what the student will focus on in upcoming lessons. For example: 'Focus on pronunciation and fluency in professional contexts'. This is set by the teacher after each lesson.
Caveat	Uses the most recent lesson's teacher feedback. NULL values are excluded. This guides the next lesson's content.

Index	2.5
Section & Heading	Student Focus Request
Homepage Component	Student Focus Request (Free-text input)
Database Tables	student_class_queries.query_text
SQL Query	`SELECT scq.student_id, scq.class_id, scq.query_text FROM student_class_queries scq WHERE scq.student_id = {{student_id}} ORDER BY scq.created_at DESC LIMIT 1;`

Output	Shows what the student specifically requested to work on. For example: 'I want to focus more on telephone conversations and email writing for work'. This is the student's own input.
Caveat	Shows most recent request only. This is optional student input - may be NULL if student hasn't made any specific requests.

Section 3: Skill Snapshot (Progress within Level)

This section shows detailed progress in specific English skills: Grammar, Vocabulary, Speaking, and Pronunciation.

Index	3.0
Section & Heading	Skill Snapshot
Homepage Component	Grammar Progress (Percentage layer)
Database Tables	level_assessments.grammar_score, lesson_feedbacks.grammar_rate
SQL Query	`SELECT {{student_id}} AS student_id, la.grammar_score, lf.grammar_rate, ROUND(COALESCE(la.grammar_score, 0) * 0.6 + COALESCE(lf.grammar_rate * 20, 0) * 0.4) AS grammar_progress FROM (SELECT student_id, grammar_score FROM level_assessments WHERE student_id = {{student_id}} ORDER BY assessed_at DESC LIMIT 1) la LEFT JOIN (SELECT student_id, AVG(grammar_rate) AS`

	`grammar_rate FROM lesson_feedbacks WHERE student_id = {{student_id}} AND grammar_rate IS NOT NULL GROUP BY student_id) lf ON la.student_id = lf.student_id;`
Output	Shows grammar progress as a percentage (0-100%). Combines assessment score (60% weight) with average lesson feedback (40% weight). For example: '78% - Good progress in grammar'.
Caveat	Weighted calculation: Assessment score × 0.6 + Average lesson feedback × 0.4. Assessment scores are 0-100, lesson feedback is 1-5 (multiplied by 20 for percentage).

Index	3.1
Section & Heading	Vocabulary Progress
Homepage Component	Vocabulary Progress (Percentage layer)
Database Tables	level_assessments.vocabulary_score, student_progress.vocabulary_mastered
SQL Query	`SELECT sp.student_id, la.vocabulary_score, sp.vocabulary_mastered, ROUND(COALESCE(la.vocabulary_score, 0) * 0.5 + LEAST(sp.vocabulary_mastered / 200.0 * 100, 100) * 0.5) AS vocabulary_progress FROM student_progress sp LEFT JOIN (SELECT student_id, vocabulary_score FROM level_assessments WHERE student_id = {{student_id}} ORDER BY assessed_at DESC LIMIT 1) la ON sp.student_id = la.student_id WHERE sp.student_id = {{student_id}};`
Output	Shows vocabulary progress as a percentage. Combines assessment score (50% weight) with number of words mastered (50% weight). For example: '82% - Strong vocabulary base with 150 words mastered'.

Caveat	Vocabulary mastered is capped at 200 words for calculation (100%). Weighted: Assessment score × 0.5 + (Words mastered / 200) × 100 × 0.5.

Index	3.2
Section & Heading	Speaking Progress
Homepage Component	Speaking Progress (Percentage layer)
Database Tables	level_assessments.fluency_score, lesson_feedbacks.speaking_rate
SQL Query	`SELECT {{student_id}} AS student_id, la.fluency_score, lf.speaking_rate, ROUND(COALESCE(la.fluency_score, 0) * 0.6 + COALESCE(lf.speaking_rate * 20, 0) * 0.4) AS speaking_progress FROM (SELECT student_id, fluency_score FROM level_assessments WHERE student_id = {{student_id}} ORDER BY assessed_at DESC LIMIT 1) la LEFT JOIN (SELECT student_id, AVG(speaking_rate) AS speaking_rate FROM lesson_feedbacks WHERE student_id = {{student_id}} AND speaking_rate IS NOT NULL GROUP BY student_id) lf ON la.student_id = lf.student_id;`
Output	Shows speaking/fluency progress as a percentage. Combines assessment fluency score (60% weight) with average lesson speaking feedback (40% weight). For example: '85% - Good speaking fluency'.
Caveat	Same weighted calculation as grammar: Fluency assessment × 0.6 + Average speaking feedback × 0.4. Speaking feedback is 1-5 scale, multiplied by 20.

Index	3.3
Section & Heading	Pronunciation Progress
Homepage Component	Pronunciation Progress (Percentage layer)
Database Tables	lesson_feedbacks.pronunciation_rate
SQL Query	`SELECT student_id, ROUND(AVG(pronunciation_rate), 1) AS avg_pronunciation, ROUND(AVG(pronunciation_rate) * 20) AS pronunciation_progress FROM lesson_feedbacks WHERE student_id = {{student_id}} AND pronunciation_rate IS NOT NULL GROUP BY student_id;`
Output	Shows pronunciation progress based solely on teacher feedback from lessons. Returns percentage and average rating. For example: '76% - Pronunciation improving (avg 3.8/5)'.
Caveat	Only uses lesson feedback (no assessment component). Average of all pronunciation ratings from lessons, converted to percentage (rating × 20).

Section 4: Practice & Games

This section shows practice exercises and games that students can complete between lessons.

Index	4.0
Section & Heading	Practice & Games
Homepage Component	Recent Practice List (Incomplete items)
Database Tables	games.exercise_type, games.status
SQL Query	`SELECT g.id AS game_id, g.exercise_type, g.status, g.topic_name FROM games g WHERE g.student_id = {{student_id}} AND g.status IN ('pending', 'approved') ORDER BY g.created_at DESC LIMIT 5;`
Output	Shows the 5 most recent practice exercises that haven't been completed. For example: 'Vocabulary Quiz - Pending', 'Grammar Exercise - In Progress'. Students can click to continue or start these exercises.
Caveat	Only shows incomplete items (status = pending or approved). Completed exercises are hidden. Limited to 5 most recent items.

Index	4.1
Section & Heading	Lesson Reference
Homepage Component	Lesson Reference (Practice source)

Database Tables	games.class_id, classes.meeting_start
SQL Query	`SELECT g.id AS game_id, g.class_id, c.meeting_start AS class_date, u.full_name AS teacher_name FROM games g INNER JOIN classes c ON g.class_id = c.id INNER JOIN users u ON c.teacher_id = u.id WHERE g.student_id = {{student_id}} ORDER BY g.created_at DESC LIMIT 5;`
Output	Shows which lesson each practice exercise came from, including the date and teacher. Helps students remember the context. For example: 'From Jan 28 lesson with Ms. Johnson'.
Caveat	Links practice exercises back to their source lessons. Useful for students to understand where practice topics came from.

Index	4.2
Section & Heading	Estimated Time
Homepage Component	Estimated Time (Minutes)
Database Tables	adaptive_assessment_questions.average_time_seconds
SQL Query	`SELECT g.exercise_type, CEIL(COALESCE(AVG(aaq.average_time_seconds), 120) / 60) AS estimated_minutes FROM games g LEFT JOIN adaptive_assessment_questions aaq ON aaq.skill_focus = g.exercise_type WHERE g.student_id = {{student_id}} AND g.status = 'pending' GROUP BY g.exercise_type;`

Output	Shows how long each practice exercise will take. For example: '10 minutes' or '5 minutes'. Helps students plan their study time.
Caveat	Uses average completion time from past students. Default is 2 minutes (120 seconds) if no data available. Time is rounded up to nearest minute.

Section 5: Weekly Highlights

This section summarizes the student's achievements and activity over the past 7 days.

Index	5.0
Section & Heading	Weekly Highlights
Homepage Component	New Words Learned
Database Tables	student_progress.vocabulary_mastered, class_summaries.vocabulary_learned
SQL Query	`SELECT c.student_id, COALESCE(SUM(JSON_LENGTH(cs.vocabulary_learned)), sp.vocabulary_mastered) AS words_count FROM classes c LEFT JOIN class_summaries cs ON c.id = cs.class_id LEFT JOIN student_progress sp ON c.student_id = sp.student_id WHERE c.student_id = {{student_id}} AND c.status = 'ended' AND c.meeting_start >= DATE_SUB(NOW(), INTERVAL 7 DAY) GROUP BY c.student_id, sp.vocabulary_mastered;`

Output	Shows total number of new words learned this week. For example: 'Learned 24 new words this week'. Counts vocabulary from all completed lessons in the past 7 days.
Caveat	Time period is hardcoded to 7 days (INTERVAL 7 DAY). Counts words from JSON array in vocabulary_learned field. Falls back to total mastered if no recent lesson data.

Index	5.1
Section & Heading	Lessons Completed
Homepage Component	Lessons Completed (Count)
Database Tables	classes.id (Count where status = 'ended')
SQL Query	`SELECT student_id, COUNT(*) AS lessons_completed FROM classes WHERE student_id = {{student_id}} AND status = 'ended' AND meeting_start >= DATE_SUB(NOW(), INTERVAL 7 DAY) GROUP BY student_id;`
Output	Shows how many lessons the student completed this week. For example: '3 lessons completed this week'. Simple count of finished lessons.
Caveat	Time period is hardcoded to 7 days. Only counts lessons with status 'ended'. This is a motivational metric showing weekly activity.

Index	5.2
Section & Heading	Speaking Percentage
Homepage Component	Speaking Percentage (Achievement chip)
Database Tables	class_summaries.engagement_level
SQL Query	`SELECT c.student_id, cs.engagement_level, CASE cs.engagement_level WHEN 'very_high' THEN 95 WHEN 'high' THEN 85 WHEN 'medium' THEN 70 WHEN 'low' THEN 50 ELSE 60 END AS speaking_percentage FROM classes c INNER JOIN class_summaries cs ON c.id = cs.class_id WHERE c.student_id = {{student_id}} AND c.status = 'ended' ORDER BY c.meeting_start DESC LIMIT 1;`
Output	Shows student engagement as a percentage based on most recent lesson. For example: '85% engagement - High participation'. This is a quick achievement indicator.
Caveat	Engagement levels map to percentages: very_high=95%, high=85%, medium=70%, low=50%, unknown=60%. Based on teacher's assessment from most recent lesson only.

Section 6: Additional Student Features

These queries provide detailed information about past lessons and learning statistics.

Index	6.0
Section & Heading	System Adaptation Message
Homepage Component	Personalized Confidence Line
Database Tables	Not possible from existing database (Product-wide static text)
SQL Query	`N/A - Static motivational text`
Output	Displays an encouraging message to boost student confidence. For example: 'Keep up the great work! You are making excellent progress.' This is static text, not database-driven.
Caveat	Not a database query. Motivational messages are hardcoded in the application. May vary based on student progress in future versions.

Index	6.1
Section & Heading	Class Learning Summary
Homepage Component	What student learned in a class

Database Tables	classes, class_summaries, users
SQL Query	`SELECT c.id AS class_id, c.meeting_start AS class_date, u.full_name AS teacher_name, cs.summary_text, cs.topics_detected, cs.vocabulary_learned, cs.grammar_concepts, cs.strengths, cs.engagement_level FROM classes c INNER JOIN class_summaries cs ON c.id = cs.class_id INNER JOIN users u ON c.teacher_id = u.id WHERE c.student_id = {{student_id}} AND c.id = {{class_id}} ORDER BY c.meeting_start DESC;`
Output	Provides a complete summary of what the student learned in a specific lesson. Includes topics covered, new vocabulary, grammar points, strengths identified, and overall engagement. This is the detailed post-lesson report.
Caveat	Requires both {{student_id}} and {{class_id}} parameters. Remove class_id filter to show all lessons. This is comprehensive lesson feedback from the teacher.

Index	6.2
Section & Heading	Next Class Planning
Homepage Component	What teacher planned for next class
Database Tables	classes, class_summaries
SQL Query	`SELECT c.id AS class_id, c.meeting_start AS class_date, cs.areas_for_improvement AS next_class_focus, c.student_goal, c.student_goal_note FROM classes c LEFT JOIN class_summaries cs`

	`ON c.id = cs.class_id WHERE c.student_id = {{student_id}} AND c.status = 'ended' ORDER BY c.meeting_start DESC LIMIT 1;`
Output	Shows what the teacher has planned for the next lesson based on the most recent completed lesson. Includes areas for improvement and student's goals. Helps students prepare for upcoming lessons.
Caveat	Uses most recent ended lesson's feedback. Shows teacher's recommendations for next focus areas. This bridges past and future lessons.

Index	6.3
Section & Heading	Words Per Class
Homepage Component	How many new words students learn in EACH CLASS
Database Tables	classes, class_summaries, users
SQL Query	`SELECT c.id AS class_id, c.meeting_start AS class_date, u.full_name AS teacher_name, COALESCE(JSON_LENGTH(cs.vocabulary_learned), 0) AS words_learned FROM classes c INNER JOIN users u ON c.teacher_id = u.id LEFT JOIN class_summaries cs ON c.id = cs.class_id WHERE c.student_id = {{student_id}} AND c.status = 'ended' ORDER BY c.meeting_start DESC;`
Output	Shows vocabulary progress per lesson. For example: 'Jan 28 - 12 words, Jan 25 - 8 words'. This creates a per-lesson vocabulary learning chart. Helps students see their learning pattern across individual classes.

Caveat	Returns all completed lessons ordered by date. Word count comes from vocabulary_learned JSON array length. Zero if no vocabulary data recorded.

Index	6.4
Section & Heading	Words Per Week
Homepage Component	How many words students learn PER WEEK
Database Tables	classes, class_summaries
SQL Query	`SELECT c.student_id, DATE(DATE_SUB(c.meeting_start, INTERVAL WEEKDAY(c.meeting_start) DAY)) AS week_start, DATE(DATE_ADD(DATE_SUB(c.meeting_start, INTERVAL WEEKDAY(c.meeting_start) DAY), INTERVAL 6 DAY)) AS week_end, SUM(COALESCE(JSON_LENGTH(cs.vocabulary_learned), 0)) AS words_learned, COUNT(c.id) AS classes_count FROM classes c LEFT JOIN class_summaries cs ON c.id = cs.class_id WHERE c.student_id = {{student_id}} AND c.status = 'ended' GROUP BY c.student_id, week_start, week_end ORDER BY week_start DESC LIMIT 12;`
Output	Shows vocabulary learning trends by week. For example: 'Week of Jan 22: 28 words in 3 classes'. This creates a weekly vocabulary chart showing learning consistency over 12 weeks.
Caveat	Limited to last 12 weeks. Groups by calendar week (Monday-Sunday). Includes lesson count to show activity level. Useful for identifying study patterns.

Section 7: Teacher Analytics

This section provides analytics for teachers to track student engagement and retention.

Index	7.0
Section & Heading	Teacher Analytics
Homepage Component	Teacher Churn Analysis
Database Tables	classes.teacher_id, classes.student_id, classes.is_present, users.full_name
SQL Query	`SELECT c.teacher_id, u.full_name AS teacher_name, COUNT(DISTINCT c.student_id) AS total_students_taught, COUNT(DISTINCT CASE WHEN c.is_present = 0 THEN c.student_id END) AS students_absent, COUNT(DISTINCT CASE WHEN NOT EXISTS (SELECT 1 FROM classes c2 WHERE c2.student_id = c.student_id AND c2.teacher_id = c.teacher_id AND c2.meeting_start > DATE_ADD(NOW(), INTERVAL -30 DAY)) THEN c.student_id END) AS churned_students FROM classes c INNER JOIN users u ON c.teacher_id = u.id WHERE c.teacher_id = {{teacher_id}} AND c.meeting_start BETWEEN {{start_date}} AND {{end_date}} AND c.status = 'ended' GROUP BY c.teacher_id, u.full_name;`
Output	Shows teacher retention metrics. Counts: (1) Total unique students taught, (2) Students who didn't show up, (3) Students who haven't returned in 30 days (churned). For example: 'Taught 45 students, 5 absent, 8 churned'.
Caveat	Requires 3 parameters: {{teacher_id}}, {{start_date}}, {{end_date}}. Churn period is hardcoded to 30 days - students who haven't had a lesson in last 30 days are considered churned. This helps teachers track retention.

Important Notes

Parameter Requirements

Most queries require {{student_id}} parameter to be passed from the API. This ensures each student only sees their own data. The API must retrieve this from the authenticated user's session.

Hardcoded Values

Several queries have hardcoded time periods and thresholds:

• Join lesson button: 5 minutes before lesson (can be changed via INTERVAL)

• Cancellation policy: 24 hours before lesson (TIMESTAMPDIFF threshold)

• Weekly highlights: 7 days (DATE_SUB INTERVAL 7 DAY)

• Teacher churn: 30 days without lesson (INTERVAL -30 DAY)

• Vocabulary mastery cap: 200 words (can be adjusted)

Data Quality

Many queries use COALESCE or NULL checks to handle missing data gracefully. If critical data is missing (e.g., class summaries, lesson feedback), some features may show default values or be hidden from the UI.

Performance Considerations

Most queries use proper indexing (student_id, class_id, meeting_start) and include LIMIT clauses to ensure fast performance. Queries returning multiple weeks of data (6.4) are limited to 12 weeks to prevent performance issues.

— End of Documentation —