

**Milestone: Implementation in MySQL**  
**Project: Personalized Period Tracker with Analytics and Social Sync**

Group 8  
Hanisha Reddy Cattamanchi Gopinath  
Sakshi Pawar

+1 848-256-8721  
+1 617-447-8235

Cattamanchigopinat.h@northeastern.edu  
pawar.saks@northeastern.edu

**Percentage of Effort Contributed by Student1: 50%**

**Percentage of Effort Contributed by Student2: 50%**

**Signature of Student 1:**



**Signature of Student 2:**



**Submission Date: 11-09-2025**

## INTRODUCTION:

The **Personalized Period Tracker Database** is designed to help users log and analyze their menstrual health data, track symptoms, manage cycle records, and connect socially through synchronized cycle insights. This database organizes user information, period and symptom logs, reminders, and analytical insights in a structured way, enabling accurate predictions and meaningful health trends over time.

For this project, we designed a relational database in MySQL based on the logical data model and populated it with realistic random data to simulate user interactions. The dataset maintains referential integrity, foreign key consistency, and balanced data distributions across all entities to reflect real-world tracking scenarios. Random data was generated and inserted into the database through SQL scripts.

### 1. MySQL database:

```
CREATE DATABASE period_tracker_db;

USE period_tracker_db;

-- USER TABLE
CREATE TABLE User (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    username VARCHAR(50) NOT NULL UNIQUE,
    date_of_birth DATE NOT NULL,
    weight DECIMAL(5,2),
    height DECIMAL(5,2),
    active BOOLEAN NOT NULL
);

-- FRIENDSHIP TABLE
CREATE TABLE Friendship (
    friendship_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id_1 INT NOT NULL,
    user_id_2 INT NOT NULL,
    status ENUM('Pending','Active','Blocked') NOT NULL,
    initiated_by INT NOT NULL,
    FOREIGN KEY (user_id_1) REFERENCES User(user_id),
    FOREIGN KEY (user_id_2) REFERENCES User(user_id),
    FOREIGN KEY (initiated_by) REFERENCES User(user_id)
);
```

```

-- SYNC_OVERLAP TABLE
CREATE TABLE Sync_Overlap (
    sync_id INT AUTO_INCREMENT,
    friendship_id INT NOT NULL,
    overlap_start DATE NOT NULL,
    overlap_end DATE NOT NULL,
    PRIMARY KEY (sync_id, friendship_id),
    FOREIGN KEY (friendship_id) REFERENCES Friendship(friendship_id)
);

-- MEDICATION_LOG TABLE
CREATE TABLE Medication_Log (
    medication_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    medication_name VARCHAR(100) NOT NULL,
    med_date DATE NOT NULL,
    dosage VARCHAR(50),
    purpose VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

-- FLOW TABLE
CREATE TABLE Flow (
    flow_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    flow_type ENUM('Light','Medium','Heavy','Spotting') NOT NULL,
    entry_date DATE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

-- SYMPTOMS TABLE
CREATE TABLE Symptoms (
    symptom_id INT AUTO_INCREMENT,
    user_id INT NOT NULL,
    symptom_type ENUM('Cramps','Headache','Bloating','Mood Swings','Fatigue','Acne','Back Pain','Nausea','Hot Flashes','Insomnia','Anxiety','Depression') NOT NULL,
    symptom_date DATE NOT NULL,
    severity ENUM('Mild','Moderate','Severe') NOT NULL,
    symptom_notes TEXT,
    PRIMARY KEY (symptom_id, user_id),
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

-- SYMPTOM_ANALYTICS TABLE
CREATE TABLE Symptom_Analytics (
    symptom_analytics_id INT AUTO_INCREMENT,
    user_id INT NOT NULL,

```

```
symptom_type ENUM('Cramps','Headache','Bloating','Mood Swings','Fatigue','Acne','Back Pain','Nausea','Hot Flashes','Insomnia','Anxiety','Depression') NOT NULL,
frequency_count INT DEFAULT 0 NOT NULL,
avg_severity DECIMAL(3,2),
most_common_cycleday INT,
PRIMARY KEY (symptom_analytics_id, user_id),
FOREIGN KEY (user_id) REFERENCES User(user_id)
);
```

-- PERIOD\_RECORD TABLE

```
CREATE TABLE Period_Record (
record_id INT AUTO_INCREMENT,
user_id INT NOT NULL,
start_date DATE NOT NULL,
end_date DATE,
PRIMARY KEY (record_id, user_id),
FOREIGN KEY (user_id) REFERENCES User(user_id),
CHECK (end_date IS NULL OR end_date >= start_date)
);
```

-- CYCLES TABLE

```
CREATE TABLE Cycles (
cycle_id INT AUTO_INCREMENT,
record_id INT NOT NULL,
user_id INT NOT NULL,
cycle_len INT NOT NULL,
period_len INT NOT NULL,
PRIMARY KEY (cycle_id, record_id),
FOREIGN KEY (record_id, user_id) REFERENCES Period_Record(record_id, user_id),
CHECK (cycle_len > 0 AND period_len > 0)
);
```

-- ANALYTICS (Superclass)

```
CREATE TABLE Analytics (
analytics_id INT AUTO_INCREMENT PRIMARY KEY,
cycle_id INT NOT NULL,
record_id INT NOT NULL,
analytics_type ENUM('Phase','Period') NOT NULL,
FOREIGN KEY (cycle_id, record_id) REFERENCES Cycles(cycle_id,record_id)
);
```

-- PHASE\_ANALYTICS (Subclass)

```
CREATE TABLE Phase_Analytics (
analytics_id INT,
period_analytics_id INT AUTO_INCREMENT PRIMARY KEY,
phase ENUM('Menstrual','Follicular','Ovulation','Luteal') NOT NULL,
```

```

phase_sdate DATE NOT NULL,
phase_edate DATE,
FOREIGN KEY (analytics_id) REFERENCES Analytics(analytics_id),
CHECK (phase_edate IS NULL OR phase_edate >= phase_sdate)
);

-- PERIOD_ANALYTICS (Subclass)
CREATE TABLE Period_Analytics (
    analytics_id INT,
    phase_analytics_id INT AUTO_INCREMENT PRIMARY KEY,
    month
    ENUM('January','February','March','April','May','June','July','August','September','October','November','December') NOT NULL,
    pred_start DATE NOT NULL,
    pred_end DATE NOT NULL,
    avg_cycle_len INT NOT NULL,
    avg_period_len INT NOT NULL,
    FOREIGN KEY (analytics_id) REFERENCES Analytics(analytics_id),
    CHECK (pred_end >= pred_start),
    CHECK (avg_cycle_len > 0 AND avg_period_len > 0)
);

-- REMINDER (Superclass)
CREATE TABLE Reminder (
    reminder_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    reminder_type ENUM('Phase','Period') NOT NULL,
    reminder_active BOOLEAN NOT NULL,
    FOREIGN KEY (user_id) REFERENCES User(user_id)
);

-- PHASE_REMINDER (Subclass)
CREATE TABLE Phase_Reminder (
    reminder_id INT PRIMARY KEY,
    notify_on_start BOOLEAN NOT NULL,
    phase ENUM('Menstrual','Follicular','Ovulation','Luteal') NOT NULL,
    FOREIGN KEY (reminder_id) REFERENCES Reminder(reminder_id)
);

-- PERIOD_REMINDER (Subclass)
CREATE TABLE Period_Reminder (
    reminder_id INT PRIMARY KEY,
    notify_on_start BOOLEAN NOT NULL,
    days_before_predicted INT CHECK (days_before_predicted >= 0),
    FOREIGN KEY (reminder_id) REFERENCES Reminder(reminder_id)
);

```

## 2. Generating Random data and uploading it in the database

The screenshot shows the MySQL Workbench interface. A query window at the top contains the SQL command: `SELECT * FROM period_tracker_db.Analytics;`. Below the query window is a result grid titled "Analytics 1" displaying the data from the Analytics table. The table has four columns: analytics\_id, cycle\_id, record\_id, and analytics\_type. The data consists of 30 rows of random integers. To the right of the result grid is a vertical toolbar with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

analytics_id	cycle_id	record_id	analytics_type
3	6	4	Period
4	7	5	Period
5	8	7	Period
6	9	9	Period
7	10	13	Period
8	11	15	Period
9	12	16	Period
10	13	19	Period
11	14	20	Period
12	15	22	Period
13	16	25	Period
14	17	27	Period
15	18	33	Period
16	19	36	Period
17	20	37	Period
18	21	40	Period
19	22	43	Period
20	23	44	Period
21	24	45	Period
22	25	47	Period
23	26	49	Period
24	27	50	Period
25	28	54	Period
26	29	56	Period

The screenshot shows the MySQL Workbench interface. A query window at the top contains the SQL command: `SELECT * FROM period_tracker_db.Cycles;`. Below the query window is a result grid titled "Cycles 1" displaying the data from the Cycles table. The table has five columns: cycle\_id, record\_id, user\_id, cycle\_len, and period\_len. The data consists of 30 rows of random integers. To the right of the result grid is a vertical toolbar with icons for Result Grid, Form Editor, Field Types, Query Stats, and Execution Plan.

cycle_id	record_id	user_id	cycle_len	period_len
6	4	2	42	5
7	5	2	16	4
8	7	3	46	4
9	9	3	33	6
10	13	5	35	4
11	15	6	28	7
12	16	6	58	6
13	19	7	26	6
14	20	7	40	5
15	22	8	27	4
16	25	9	31	6
17	27	10	41	7
18	33	13	27	7
19	36	13	1	4
20	37	14	26	6
21	40	15	34	6
22	43	16	35	4
23	44	16	27	5
24	45	16	25	6
25	47	17	31	6
26	49	18	41	7
27	50	18	43	4
28	54	20	32	5
29	56	21	30	7

1 • `SELECT * FROM period_tracker_db.Flow;`

100% 23:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

flow_id	user_id	flow_type	entry_date
3	5	Medium	2024-02-12
4	8	Light	2024-03-01
5	8	Medium	2024-03-02
6	8	Heavy	2024-03-03
7	12	Heavy	2024-03-14
8	12	Medium	2024-03-15
9	12	Light	2024-03-16
10	15	Spotting	2024-03-20
11	15	Medium	2024-03-21
12	19	Heavy	2024-03-22
13	22	Medium	2024-04-03
14	22	Heavy	2024-04-04
15	25	Light	2024-04-05
16	28	Medium	2024-04-06
17	31	Heavy	2024-04-07
18	31	Medium	2024-04-08
19	34	Light	2024-04-09
20	36	Medium	2024-04-10
21	40	Heavy	2024-04-11
22	40	Medium	2024-04-12
23	43	Light	2024-04-13
24	46	Spotting	2024-05-01
25	48	Medium	2024-05-02
26	48	Heavy	2024-05-03

Flow 1 Apply Revert

1 • `SELECT * FROM period_tracker_db.Friendship;`

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

friendship_id	user_id_1	user_id_2	status	initiated_on
3	3	7	Active	7
4	4	9	Blocked	4
5	6	11	Active	6
6	10	12	Active	10
7	13	17	Pending	13
8	14	18	Active	14
9	15	20	Blocked	15
10	16	19	Active	19
11	21	25	Pending	25
12	22	26	Active	22
13	23	27	Active	23
14	24	29	Blocked	24
15	28	30	Active	30
16	31	33	Pending	31
17	32	35	Active	35
18	34	37	Active	34
19	36	38	Blocked	38
20	39	42	Active	39
21	40	43	Pending	43
22	41	44	Active	41
23	45	47	Active	45
24	46	49	Pending	46
25	48	50	Blocked	50
26	51	53	Active	51

Friendship 1 Apply Revert

1 • `SELECT * FROM period_tracker_db.User;`

100% | 1:1

Result Grid | Filter Rows: Search | Edit: | Export/Import: |

1 User 1

Apply | Revert

user_id	first_name	last_name	email	password	username	date_of_birth	weight	height	active
3	Sneha	Kumar	sneha.kumar@example.com	hashed_pw3	snehak	2000-01-27	58.30	158.00	1
4	Tanya	Reddy	tanya.reddy@example.com	hashed_pw4	tanyareddy	1997-07-23	61.50	165.00	1
5	Neha	Pillai	neha.pillai@example.com	hashed_pw5	nehap	1998-10-09	54.60	159.00	1
6	Meena	Gupta	meena.gupta@example.com	hashed_pw6	meenag	1992-12-04	63.40	167.00	0
7	Aditi	Nair	aditi.nair@example.com	hashed_pw7	aditnair	1997-02-28	57.80	161.00	1
8	Divya	Iyer	divya.iyer@example.com	hashed_pw8	divyaiyer	1998-07-04	59.20	164.00	1
9	Pooja	Mishra	pooja.mishra@example.com	hashed_pw9	poojam	1997-09-30	62.10	163.00	1
10	Simran	Das	simran.das@example.com	hashed_pw10	simrand	2001-04-09	56.70	160.00	1
11	Kavya	Menon	kavya.menon@example.com	hashed_pw11	kavyame...	1996-11-11	58.90	162.00	1
12	Ananya	Rao	ananya.rao@example.com	hashed_pw12	ananyara...	1995-06-05	59.80	165.00	1
13	Isha	Bose	isha.bose@example.com	hashed_pw13	ishab	1999-01-20	52.40	158.00	1
14	Sakshi	Chopra	sakshi.chopra@example.com	hashed_pw14	sakshic	1998-04-15	63.00	166.00	1
15	Aarohi	Bhattacharya	aarohi.bhattacharya@exa...	hashed_pw15	aaronhb	2000-12-09	55.70	160.00	1
16	Nisha	Pandey	nisha.pandey@example.com	hashed_pw16	nishap	1994-09-18	64.50	170.00	1
17	Shruti	Desai	shruti.desai@example.com	hashed_pw17	shрудि	1997-03-07	58.10	164.00	1
18	Mitali	Chatterjee	mitali.chatterjee@example.com	hashed_pw18	mitalic	1996-05-28	60.90	163.00	1
19	Charvi	Shetty	charvi.shetty@example.com	hashed_pw19	charvis	1999-07-16	57.30	161.00	1
20	Rekha	Raman	rekha.raman@example.com	hashed_pw20	rekhar	1993-02-02	66.20	168.00	0
21	Aishwarya	Joshi	aishwarya.joshi@example....	hashed_pw21	aishwaryaj	1997-06-11	59.40	164.00	1
22	Tanvi	Goswami	tanvi.goswami@example.c...	hashed_pw22	tanvig	1999-10-03	56.80	161.00	1
23	Dia	Mukherjee	dia.mukherjee@example.com	hashed_pw23	diarmukhe...	1998-02-14	55.90	160.00	1
24	Sanya	Kaur	sanya.kaur@example.com	hashed_pw24	sanyak	1996-07-21	60.10	165.00	1
25	Rachna	Bhatagat	rachna.bhatagat@example.c...	hashed_pw25	rachnab	1993-09-19	62.40	167.00	1
26	Ira	Dey	ira.dey@example.com	hashed_pw26	irad	2000-05-28	54.10	158.00	1
27	Mukundan	Chouban	mukundan.chouban@example.com	hashed_pw27	mukundana	2001-01-12	57.60	162.00	1

1 • `SELECT * FROM period_tracker_db.Medication_Log;`

100% | 1:1

Result Grid | Filter Rows: Search | Edit: | Export/Import: |

1 Medication\_Log 1

Apply | Revert

medication_id	user_id	medication_name	med_date	dosage	purpose
3	12	Tranexamic Acid	2024-03-12	500mg twice daily	Heavy menstrual bleeding
4	15	Iron Supplement	2024-03-15	1 capsule daily	Anemia prevention
5	19	Meftal-Spas	2024-03-16	1 tablet twice a day	Cramps
6	22	Cyclopam	2024-04-02	1 tablet twice a day	Abdominal pain relief
7	25	Dolo 650	2024-04-02	NULL	Menstrual pain
8	28	Tranexamic Acid	2024-04-03	500mg	NULL
9	31	Iron Supplement	2024-04-05	1 capsule daily	Iron deficiency
10	34	Meftal-Spas	2024-04-05	1 tablet every 6 hours	Cramps
11	36	Mefenamic Acid	2024-04-08	500mg every 8 hours	Menstrual pain relief
12	40	Dolo 650	2024-04-10	1 tablet twice a day	NULL
13	43	Feronia XT	2024-04-10	1 tablet daily	Iron supplement during peri...
14	46	Tranexamic Acid	2024-04-12	NULL	Heavy bleeding
15	48	Meftal-Spas	2024-05-05	1 tablet twice daily	Cramps relief
16	51	Dolo 650	2024-05-06	1 tablet as needed	Fever and pain
17	54	Cyclopam	2024-05-07	1 tablet every 8 hours	Abdominal cramps
18	58	Meftal-Spas	2024-05-08	NULL	Pain relief
19	61	Iron Supplement	2024-05-10	1 capsule daily	Post-period fatigue
20	65	Dolo 650	2024-05-10	1 tablet twice daily	Pain management
21	67	Mefenamic Acid	2024-05-11	500mg twice a day	Menstrual pain
22	69	Feronia XT	2024-05-15	NULL	Iron levels
23	72	Meftal-Spas	2024-06-01	1 tablet twice daily	Severe cramps
24	75	Tranexamic Acid	2024-06-01	500mg morning and...	NULL
25	77	Iron Supplement	2024-06-05	1 capsule daily	Anemia management
26	80	Dolo 650	2024-06-10	1 tablet as needed	Period pain
27	82	Meftal-Spas	2024-06-10	1 tablet twice daily	Pain management

1 • `SELECT * FROM period_tracker_db.Period_Analytics;`

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

period_analytics_id	analytics_id	month	pred_start	pred_end	avg_cycle_len	avg_period_l...
3	3	January	2025-02-15	2025-03-07	42	5
4	4	February	2025-03-03	2025-03-07	16	4
5	5	January	2025-03-12	2025-03-16	46	4
6	6	March	2025-04-22	2025-04-28	33	6
7	7	January	2025-03-02	2025-03-06	35	4
8	8	January	2025-02-18	2025-02-25	28	7
9	9	February	2025-04-17	2025-04-23	58	6
10	10	January	2025-02-12	2025-02-18	26	6
11	11	February	2025-03-24	2025-03-29	40	5
12	12	January	2025-02-17	2025-02-21	27	4
13	13	January	2025-02-13	2025-02-19	31	6
14	14	January	2025-03-08	2025-03-13	41	7
15	15	January	2025-02-11	2025-02-18	27	7
16	16	April	2025-04-07	2025-04-11	1	4
17	17	January	2025-02-13	2025-02-19	26	6
18	18	January	2025-02-16	2025-02-22	34	6
19	19	January	2025-02-12	2025-02-16	35	4
20	20	February	2025-03-11	2025-03-16	27	5
21	21	March	2025-04-05	2025-04-11	25	6
22	22	January	2025-02-24	2025-03-02	31	6
23	23	January	2025-02-23	2025-03-02	41	7
24	24	February	2025-04-07	2025-04-11	43	4
25	25	January	2025-02-24	2025-03-01	32	5
26	26	January	2025-02-25	2025-03-04	30	7
27	27	February	2025-04-23	2025-04-27	55	6

Period\_Analytics 1 Apply Revert

1 • `SELECT * FROM period_tracker_db.Period_Record;`

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

record_id	user_id	start_date	end_date
3	1	2025-03-13	2025-03-19
4	2	2025-01-04	2025-01-09
5	2	2025-02-15	2025-02-19
6	2	2025-03-03	2025-03-07
7	3	2025-01-25	2025-01-29
8	3	2025-03-12	NULL
9	3	2025-03-20	2025-03-26
10	3	2025-04-22	2025-04-28
11	4	2025-01-21	NULL
12	4	2025-03-11	2025-03-15
13	5	2025-01-28	2025-01-30
14	5	2025-03-02	NULL
15	6	2025-01-21	2025-01-28
16	6	2025-02-18	2025-02-24
17	6	2025-04-17	NULL
18	6	2025-06-20	2025-06-26
19	7	2025-01-17	2025-01-23
20	7	2025-02-12	2025-02-17
21	7	2025-03-24	2025-03-31
22	8	2025-01-21	2025-01-25
23	8	2025-02-17	NULL
24	8	2025-04-27	2025-05-01
25	9	2025-01-13	2025-01-19
26	9	2025-02-13	2025-02-19
27	10	2025-04-24	2025-04-24

Period\_Record 1 Apply Revert

1 • `SELECT * FROM period_tracker_db.Phase_Analytics;`

phase_analytics_id	analytics_id	phase	phase_startdate	phase_enddate
3	256	Follicular	2025-01-17	2025-01-24
4	256	Menstrual	2025-01-12	2025-01-16
5	257	Luteal	2025-03-01	2025-03-12
6	257	Ovulation	2025-02-27	2025-02-28
7	257	Follicular	2025-02-19	2025-02-26
8	257	Menstrual	2025-02-14	2025-02-18
9	258	Luteal	2025-01-19	2025-02-14
10	258	Ovulation	2025-01-17	2025-01-18
11	258	Follicular	2025-01-09	2025-01-16
12	258	Menstrual	2025-01-04	2025-01-08
13	259	Luteal	2025-03-02	2025-03-02
14	259	Ovulation	2025-02-28	2025-03-01
15	259	Follicular	2025-02-20	2025-02-27
16	259	Menstrual	2025-02-15	2025-02-19
17	260	Luteal	2025-02-09	2025-03-11
18	260	Ovulation	2025-02-07	2025-02-08
19	260	Follicular	2025-01-30	2025-02-06
20	260	Menstrual	2025-01-25	2025-01-29
21	261	Luteal	2025-04-04	2025-04-21
22	261	Ovulation	2025-04-02	2025-04-03
23	261	Follicular	2025-03-25	2025-04-01
24	261	Menstrual	2025-03-20	2025-03-24
25	262	Luteal	2025-02-10	2025-03-01
26	262	Ovulation	2025-02-08	2025-02-09
27	262	Follicular	2025-01-31	2025-02-07

Phase\_Analytics 1 | [ Apply ] [ Revert ]

1 • `SELECT * FROM period_tracker_db.Phase_Reminder;`

reminder_id	notify_on_start	phase
1	1	Follicular
3	1	Ovulation
5	0	Ovulation
7	0	Luteal
9	1	Menstrual
11	1	Menstrual
13	0	Luteal
15	1	Ovulation
17	1	Follicular
19	0	Luteal
21	1	Menstrual
23	1	Ovulation
25	0	Follicular
27	1	Menstrual
29	1	Luteal
31	0	Ovulation
33	1	Follicular
35	1	Menstrual
37	1	Luteal
39	0	Follicular
HULL	HULL	HULL

Phase\_Reminder 1 | [ Apply ] [ Revert ]

1 • `SELECT * FROM period_tracker_db.Reminder;`

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

reminder_id	user_id	reminder_type	reminder_action
3	8	Phase	1
4	12	Period	1
5	15	Phase	0
6	19	Period	1
7	22	Phase	1
8	25	Period	0
9	31	Phase	1
10	36	Period	1
11	3	Phase	1
12	7	Period	1
13	10	Phase	1
14	14	Period	0
15	18	Phase	1
16	21	Period	1
17	27	Phase	1
18	33	Period	1
19	39	Phase	0
20	42	Period	1
21	48	Phase	1
22	52	Period	1
23	56	Phase	1
24	61	Period	1
25	67	Phase	0
26	72	Period	1
27	79	Phase	1

Reminder 1 Apply Revert

1 • `SELECT * FROM period_tracker_db.Symptom_Analytics;`

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

symptom_analytics_id	user_id	symptom_type	frequency_count	avg_severity	most_common_cycled
3	12	Bloating	6	1.50	4
4	15	Mood Swings	9	2.20	2
5	19	Fatigue	8	2.10	5
6	22	Acne	5	1.60	6
7	25	Back Pain	7	2.30	3
8	28	Nausea	4	1.40	2
9	31	Hot Flashes	3	1.90	7
10	34	Insomnia	6	2.00	8
11	36	Anxiety	8	2.40	3
12	40	Depression	5	2.10	5
13	43	Cramps	9	2.60	1
14	46	Headache	6	1.80	2
15	48	Bloating	7	1.70	4
16	51	Mood Swings	8	2.50	3
17	54	Fatigue	6	2.00	4
18	58	Acne	5	1.40	6
19	61	Back Pain	7	2.10	2
20	65	Nausea	3	1.60	5
21	67	Hot Flashes	4	1.80	8
22	69	Insomnia	6	2.20	7
23	72	Anxiety	5	2.10	6
24	75	Depression	7	2.30	3
25	77	Cramps	11	2.90	1
26	80	Headache	9	1.80	2
27	82	Bloating	5	1.60	4

Symptom\_Analytics 1 Apply Revert

1 • `SELECT * FROM period_tracker_db.Symptoms;`

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

symptom_id	user_id	symptom_type	symptom_date	severity	symptom_notes
3	12	Bloating	2024-03-11	Mild	Feeling of fullness and tightness.
4	15	Mood Swings	2024-03-11	Moderate	Irritable and anxious mood since afternoon.
5	19	Fatigue	2024-03-12	Moderate	Feeling drained with low energy.
6	22	Acne	2024-03-12	Mild	Few breakouts near chin area.
7	25	Back Pain	2024-03-13	Severe	Sharp lower back pain while sitting.
8	28	Nausea	2024-03-13	Mild	NULL
9	31	Hot Flashes	2024-03-14	Moderate	Sudden warmth and sweating for a few minutes.
10	34	Insomnia	2024-03-15	Moderate	Could not sleep well last night.
11	36	Anxiety	2024-03-15	Mild	Slight restlessness before period.
12	40	Depression	2024-03-16	Severe	Low mood and tearful feeling.
13	43	Cramps	2024-04-05	Moderate	Pain manageable after medication.
14	46	Headache	2024-04-06	Mild	Slight tension headache in the evening.
15	48	Bloating	2024-04-07	Moderate	NULL
16	51	Mood Swings	2024-04-08	Moderate	Feeling emotional and irritable.
17	54	Fatigue	2024-04-08	Mild	Tired but improved after rest.
18	58	Acne	2024-04-09	Moderate	Two new pimples around jawline.
19	61	Back Pain	2024-04-10	Severe	Back ache worsening at night.
20	65	Nausea	2024-04-11	Mild	Light nausea before lunch.
21	67	Hot Flashes	2024-04-11	Moderate	Warm sensation in chest for short duration.
22	69	Insomnia	2024-04-12	Severe	Unable to sleep till 3 AM.
23	72	Anxiety	2024-04-13	Moderate	Worried about work, felt tense.
24	75	Depression	2024-04-14	Mild	NULL
25	77	Cramps	2024-05-05	Severe	Cramping with radiating pain to thighs.
26	80	Headache	2024-05-06	Moderate	Pain reduced after Dolo 650.
27	82	Bloating	2024-05-07	Mild	Abdominal pain right after lunch.

Symptoms 1 Apply Revert

1 • `SELECT * FROM period_tracker_db.Sync_Overlap;`

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

sync_id	friendship_id	overlap_start	overlap_end
3	3	2023-04-18	2023-05-10
4	68	2023-05-20	2023-06-12
5	12	2023-06-25	2023-07-15
6	49	2023-07-05	2023-07-30
7	21	2023-07-22	2023-08-10
8	34	2023-08-12	2023-09-01
9	2	2023-09-15	2023-10-05
10	50	2023-10-01	2023-10-28
11	9	2023-11-05	2023-11-25
12	37	2023-11-20	2023-12-10
13	26	2023-12-01	2023-12-20
14	64	2024-01-10	2024-01-28
15	11	2024-01-22	2024-02-15
16	30	2024-02-10	2024-03-05
17	55	2024-03-12	2024-03-28
18	17	2024-03-25	2024-04-12
19	1	2024-04-01	2024-04-25
20	63	2024-04-10	2024-04-30
21	23	2024-05-05	2024-05-22
22	40	2024-05-15	2024-06-05
23	33	2024-06-10	2024-06-28
24	18	2024-06-25	2024-07-12
25	5	2024-07-01	2024-07-20
26	45	2024-07-15	2024-08-08
27	27	2024-08-01	2024-08-25

Sync\_Overlap 1 Apply Revert

### 3. SQL queries on database

#### a. Show friends with overlapping cycles

```
SELECT
    f.user_id_1,
    f.user_id_2,
    so.overlap_start,
    so.overlap_end
FROM Sync_Overlap so
JOIN Friendship f ON so.friendship_id = f.friendship_id
WHERE f.status = 'Active';
```

The screenshot shows the MySQL Workbench interface. The SQL editor window contains the provided SQL query. The results are displayed in a grid titled 'Result Grid' with columns: user\_id\_1, user\_id\_2, overlap\_start, and overlap\_end. The results show various pairs of user IDs and their corresponding overlap dates. A sidebar on the right provides navigation links for Result Grid, Form Editor, Field Types, Query Stats, and a dropdown menu.

user_id_1	user_id_2	overlap_start	overlap_end
1	5	2024-04-01	2024-04-25
3	7	2023-04-18	2023-05-10
6	11	2024-07-01	2024-07-20
10	12	2025-08-10	2025-08-30
14	18	2025-07-01	2025-07-25
16	19	2024-09-20	2024-10-10
22	26	2023-06-25	2023-07-15
23	27	2025-05-01	2025-05-20
28	30	2023-03-10	2023-04-02
32	35	2024-03-25	2024-04-12
34	37	2024-06-25	2024-07-12
39	42	2025-01-15	2025-02-05
41	44	2025-03-05	2025-03-25
45	47	2024-05-05	2024-05-22
51	53	2023-12-01	2023-12-20
52	55	2024-08-01	2024-08-25

b. Find users with the highest number of active friends

```
SELECT
    u.user_id,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    COUNT(*) AS active_friends
FROM Friendship f
JOIN User u ON f.user_id_1 = u.user_id OR f.user_id_2 = u.user_id
WHERE f.status = 'Active'
GROUP BY u.user_id
ORDER BY active_friends DESC;
```

The screenshot shows a database query interface with the following details:

- Query Editor:** Displays the SQL code for finding users with the highest number of active friends.
- Result Grid:** Shows the output of the query, listing 14 users along with their user IDs and active friend counts.
- Toolbar:** Includes various icons for file operations, search, and export.
- Bottom Panel:** Features a "Result Grid" button, a "Read Only" status indicator, and other navigation links like "Form Editor", "Field Types", and "Query Stats".

user_id	user_name	active_friend...
30	Bhavna Taneja	2
3	Sneha Kumar	2
5	Neha Pillai	2
1	Riya Patel	2
7	Aditi Nair	2
55	Ishita Gadre	2
10	Simran Das	2
11	Kavya Menon	2
45	Aaradhyaa Patil	2
58	Ira Raj	1
118	Rajni Nair	1
18	Mitali Chatter...	1
19	Charvi Shetty	1
20	Rekha Raman	1
22	Tanvi Goswami	1
23	Dia Mukherjee	1
20	Ina Dey	1

c. Track most commonly used medications and purposes

```
SELECT
    medication_name,
    purpose,
    COUNT(*) AS usage_count
FROM Medication_Log
GROUP BY medication_name, purpose
ORDER BY usage_count DESC;
```

The screenshot shows a database query interface with the following details:

- Query Editor:** Displays the SQL code for the query.
- Result Grid:** Shows the output of the query, which is a table with three columns: medication\_name, purpose, and usage\_count. The data includes various medications like Meftal-Spas, Tranexamic Acid, Iron Supplement, etc., and their corresponding purposes and usage counts.
- Toolbar:** Includes standard database icons for file operations, search, and export.
- Right Panel:** Contains links to other tools: Result Grid, Form Editor, Field Types, and Query Stats.
- Status Bar:** Shows "Result 3" and "Read Only".

medication_na...	purpose	usage_count
Meftal-Spas	Cramps	7
Tranexamic Acid	Heavy bleeding	5
Meftal-Spas	Severe cramps	3
Iron Supplement	HULL	3
Tranexamic Acid	HULL	3
Feronia XT	Iron support	3
Tranexamic Acid	Heavy flow control	3
Dolo 650	HULL	3
Meftal-Spas	Cramps relief	3
Cyclopam	Pain relief	2
Feronia XT	Iron deficiency	2
Cyclopam	Abdominal pain	2
Mefenamic Acid	Pain control	2
Dolo 650	Pain relief	2
Dolo 650	Fever and pain	2
Cyclopam	Abdominal cramps	2

d. Find the most common symptom for each user

```
SELECT
    u.user_id,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    s.symptom_type,
    COUNT(*) AS frequency
FROM Symptoms s
JOIN User u ON s.user_id = u.user_id
GROUP BY u.user_id, s.symptom_type
HAVING COUNT(*) = (
    SELECT MAX(counts.symptom_count)
    FROM (
        SELECT user_id, symptom_type, COUNT(*) AS symptom_count
        FROM Symptoms
        GROUP BY user_id, symptom_type
    ) AS counts
    WHERE counts.user_id = s.user_id
);
```

The screenshot shows a database interface with a query editor at the top and a result grid below. The query editor contains the SQL code provided above. The result grid displays the following data:

user_id	user_name	symptom_ty...	frequency
5	Neha Pillai	Cramps	2
8	Divya Iyer	Headache	2
12	Ananya Rao	Bloating	2
15	Aarohi Bhattacharya	Mood Swings	2
19	Charvi Shetty	Fatigue	4
22	Tanvi Goswami	Acne	2
25	Rachna Bhagat	Back Pain	2
28	Avni Bansal	Nausea	2
31	Pallavi Kulkarni	Hot Flashes	2
34	Madhuri Naik	Insomnia	2
36	Ritika Rana	Anxiety	2
40	Trisha Malhotra	Depression	2

e. Average severity per symptom type (across all users)

```
SELECT
    symptom_type,
    ROUND(AVG(
        CASE
            WHEN severity = 'Mild' THEN 1
            WHEN severity = 'Moderate' THEN 2
            WHEN severity = 'Severe' THEN 3
        END
    ), 2) AS avg_severity_score
FROM Symptoms
GROUP BY symptom_type
ORDER BY avg_severity_score DESC;
```

The screenshot shows a database query editor interface with a dark theme. At the top, there's a toolbar with various icons. Below it is a code editor window displaying the SQL query. The code is numbered from 1 to 13. The result grid below shows the output of the query, which lists 14 symptom types and their average severity scores. The results are ordered by average severity score in descending order. On the right side of the interface, there's a sidebar with icons for Result Grid, Form Editor, Field Types, and other database-related functions. A status bar at the bottom indicates "Result 8".

symptom_ty...	avg_severity_sco...
Cramps	2.50
Mood Swings	2.30
Fatigue	2.20
Insomnia	2.17
Back Pain	2.13
Depression	2.00
Hot Flashes	1.67
Anxiety	1.57
Headache	1.50
Acne	1.50
Bloating	1.20
Nausea	1.00

#### f. Symptoms Correlated with Flow Type

```
SELECT
    f.flow_type,
    s.symptom_type,
    COUNT(*) AS occurrences
FROM Flow f
JOIN Symptoms s
    ON f.user_id = s.user_id
    AND f.entry_date = s.symptom_date
GROUP BY f.flow_type, s.symptom_type
ORDER BY f.flow_type, occurrences DESC;
```

The screenshot shows a database interface with a query editor at the top and a results grid below. The query editor contains the SQL code provided above. The results grid displays the output of the query, which is a list of symptoms grouped by flow type, showing their occurrence count. The results are as follows:

flow_type	symptom_ty...	occurrenc...
Light	Fatigue	1
Light	Hot Flashes	1
Medium	Cramps	1
Medium	Mood Swings	1
Medium	Acne	1
Medium	Nausea	1
Heavy	Back Pain	1

### g. Users with the Most Active Friendships

```
SELECT
    u.user_id,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    COUNT(*) AS active_friend_count
FROM Friendship f
JOIN User u ON f.user_id_1 = u.user_id OR f.user_id_2 = u.user_id
WHERE f.status = 'Active'
GROUP BY u.user_id
ORDER BY active_friend_count DESC;
```

The screenshot shows a database management interface with the following details:

- SQL Editor:** Displays the query code with line numbers 1 through 10.
- Result Grid:** Shows the output of the query in a tabular format. The columns are labeled `user_id`, `user_name`, and `active_friend_co...`. The data rows are:

user_id	user_name	active_friend_co...
30	Bhavna Taneja	2
3	Sneha Kumar	2
5	Neha Pillai	2
1	Riya Patel	2
7	Aditi Nair	2
55	Ishita Gadre	2
10	Simran Das	2
11	Kavya Menon	2
45	Aaradhy Patil	2
58	Ira Raj	1
118	Rajni Nair	1
18	Mitali Chatter...	1

- Status Bar:** Shows "100%" and "1:10".
- Toolbar:** Includes icons for file operations, search, and export.
- Right Panel:** Contains buttons for "Result Grid", "Form Editor", and "Field Types".
- Bottom Status:** Shows "Result 11" and "Read Only".

#### **h. Average Time Between Symptom Reports**

```
SELECT
    user_id,
    ROUND(AVG(days_diff), 1) AS avg_days_between_symptoms
FROM (
    SELECT
        user_id,
        DATEDIFF(symptom_date, LAG(symptom_date) OVER (PARTITION BY user_id
ORDER BY symptom_date)) AS days_diff
    FROM Symptoms
) AS sub
WHERE days_diff IS NOT NULL
GROUP BY user_id;
```

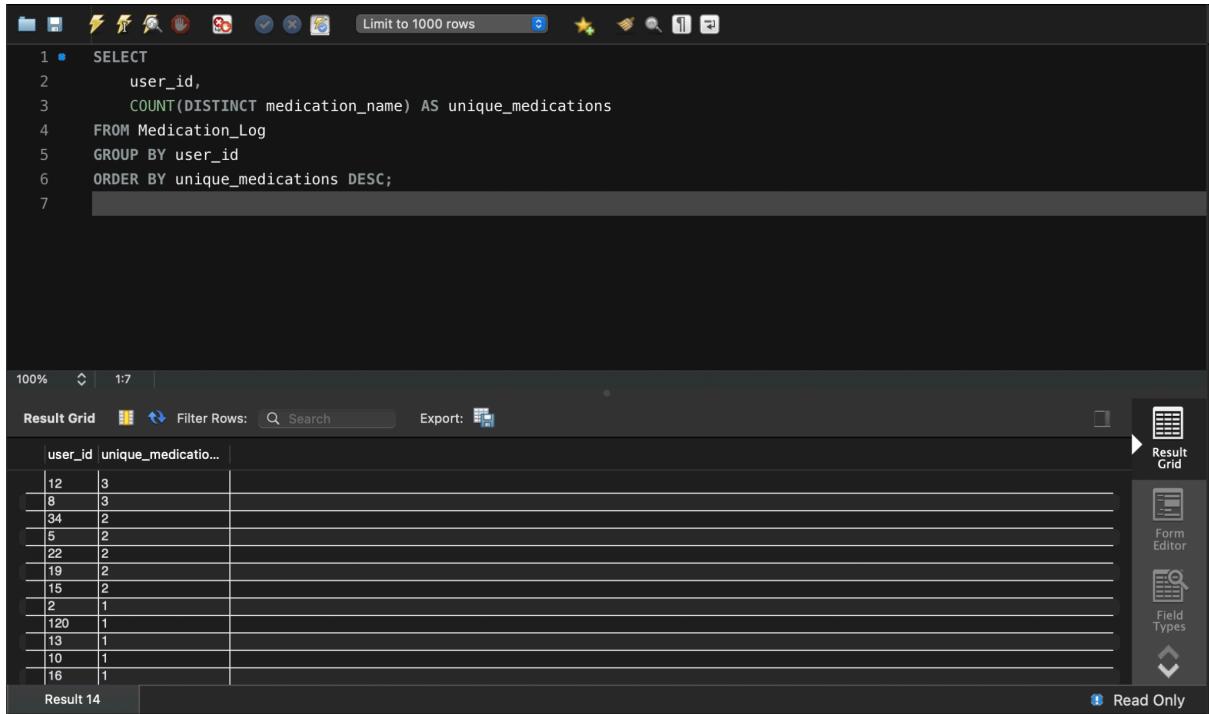
The screenshot shows a database query editor interface. At the top, there's a toolbar with various icons. Below it is a code editor window containing the SQL query provided above. The code is numbered from 1 to 12. The result grid below shows the output of the query, which consists of two columns: 'user\_id' and 'avg\_days\_between\_symptoms'. The data rows are:

user_id	avg_days_between_symptoms
3	153.0
5	89.0
8	89.3
12	89.0
15	89.3
19	89.0
22	89.3
25	134.0
28	147.0
31	146.0
34	146.0
36	147.0

At the bottom right of the result grid, there are several icons: 'Result Grid' (selected), 'Form Editor', 'Field Types', and a dropdown arrow. A 'Read Only' button is also present.

### i. Users taking the highest variety of medications

```
SELECT
    user_id,
    COUNT(DISTINCT medication_name) AS unique_medications
FROM Medication_Log
GROUP BY user_id
ORDER BY unique_medications DESC;
```



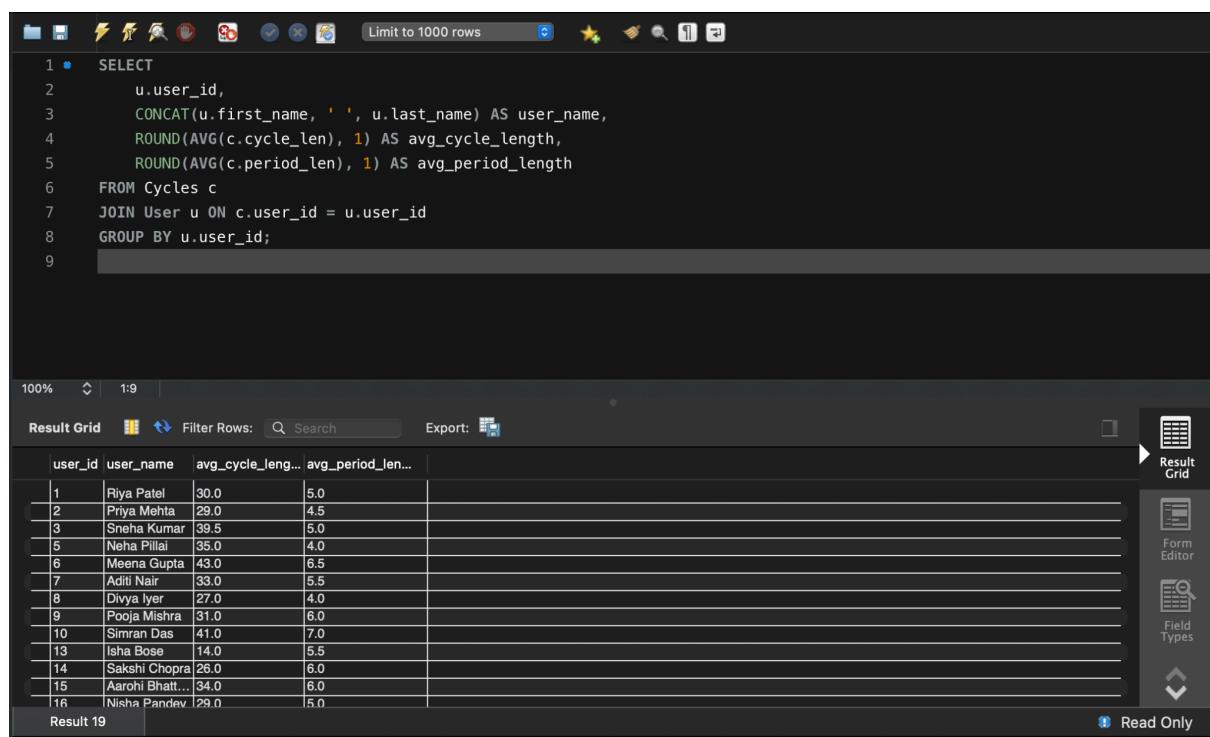
The screenshot shows a database query interface with the following details:

- Query Editor:** Displays the SQL code with line numbers 1 through 7.
- Result Grid:** Shows the output of the query in a tabular format. The columns are labeled "user\_id" and "unique\_medicatio...". The data rows are:

user_id	unique_medicatio...
12	3
8	3
34	2
5	2
22	2
19	2
15	2
2	1
120	1
13	1
10	1
16	1
- Toolbar:** Includes icons for file operations, search, and export.
- Right Panel:** Contains buttons for "Result Grid", "Form Editor", and "Field Types".
- Status Bar:** Shows "Result 14" and "Read Only".

j. Calculate each user's average cycle and period length

```
SELECT
    u.user_id,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    ROUND(AVG(c.cycle_len), 1) AS avg_cycle_length,
    ROUND(AVG(c.period_len), 1) AS avg_period_length
FROM Cycles c
JOIN User u ON c.user_id = u.user_id
GROUP BY u.user_id;
```



The screenshot shows the MySQL Workbench interface with a query editor and a result grid.

**Query Editor:**

```
1 • SELECT
2     u.user_id,
3     CONCAT(u.first_name, ' ', u.last_name) AS user_name,
4     ROUND(AVG(c.cycle_len), 1) AS avg_cycle_length,
5     ROUND(AVG(c.period_len), 1) AS avg_period_length
6 FROM Cycles c
7 JOIN User u ON c.user_id = u.user_id
8 GROUP BY u.user_id;
9
```

**Result Grid:**

	user_id	user_name	avg_cycle_length	avg_period_length
1	1	Riya Patel	30.0	5.0
2	2	Priya Mehta	29.0	4.5
3	3	Sneha Kumar	39.5	5.0
5	5	Neha Pillai	35.0	4.0
6	6	Meena Gupta	43.0	6.5
7	7	Aditi Nair	33.0	5.5
8	8	Divya Iyer	27.0	4.0
9	9	Pooja Mishra	31.0	6.0
10	10	Simran Das	41.0	7.0
13	13	Isha Bose	14.0	5.5
14	14	Sakshi Chopra	26.0	6.0
15	15	Aarohi Bhattacharya	34.0	6.0
16	16	Nisha Pandey	29.0	5.0

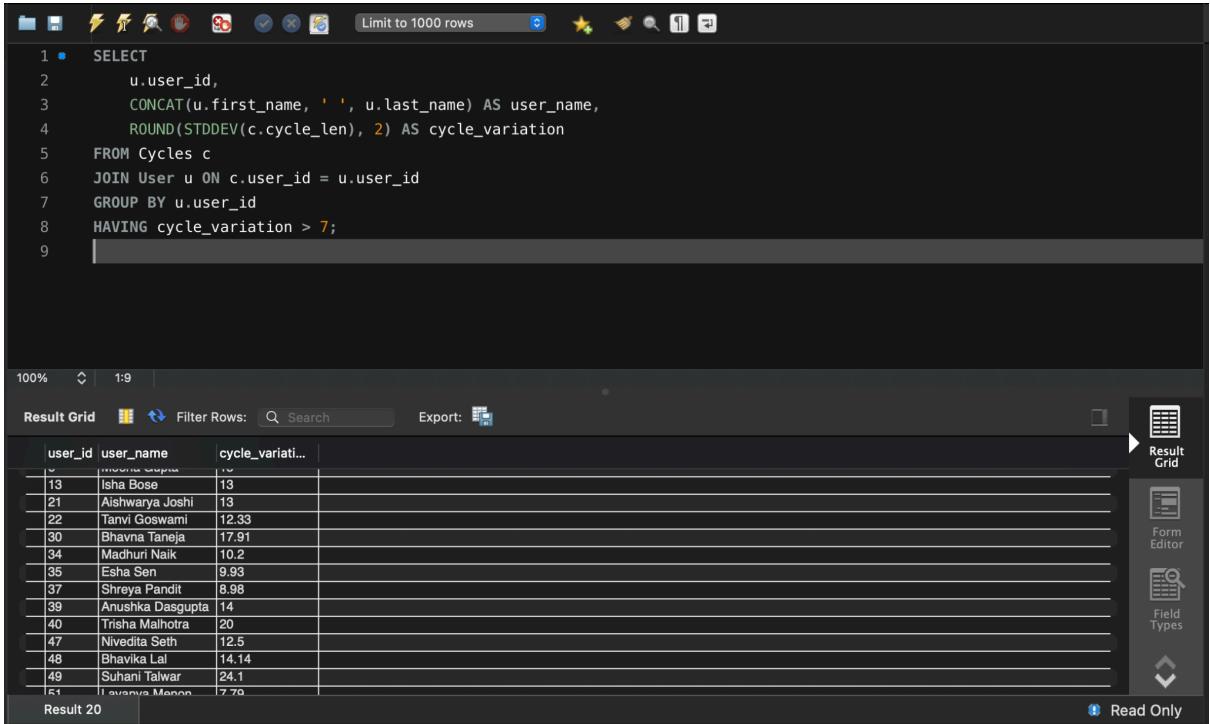
Result 19

Read Only

On the right side of the result grid, there is a sidebar with icons for Result Grid, Form Editor, and Field Types, along with a "Read Only" status indicator.

**k. Identify users with irregular cycles (variance > 7 days)**

```
SELECT
    u.user_id,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    ROUND(STDDEV(c.cycle_len), 2) AS cycle_variation
FROM Cycles c
JOIN User u ON c.user_id = u.user_id
GROUP BY u.user_id
HAVING cycle_variation > 7;
```



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Displays the SQL query from step k.
- Result Grid:** Shows the output of the query, listing 20 users with their user IDs, names, and calculated cycle variations. The variation values range from 7.70 to 24.1.
- Toolbar:** Includes standard icons for file operations, search, and export.
- Right Panel:** Contains tabs for "Result Grid" (selected), "Form Editor", and "Field Types".
- Status Bar:** Shows "Result 20" and "Read Only".

user_id	user_name	cycle_variat...
13	Isha Bose	13
21	Aishwarya Joshi	13
22	Tanvi Goswami	12.33
30	Bhavna Taneja	17.91
34	Madhuri Naik	10.2
35	Esha Sen	9.93
37	Shreya Pandit	8.98
39	Anushka Dasgupta	14
40	Trisha Malhotra	20
47	Nivedita Seth	12.5
48	Bhavika Lal	14.14
49	Suhani Talwar	24.1
51	Avneena Menon	7.70

## I. Detect missed or skipped cycles

```
SELECT
    user_id,
    record_id,
    start_date,
    end_date
FROM Period_Record
WHERE DATEDIFF(IFNULL(end_date, CURDATE()), start_date) > 40;
```

The screenshot shows the MySQL Workbench interface. The top pane displays the SQL query:

```
1 •   SELECT
2       user_id,
3       record_id,
4       start_date,
5       end_date
6   FROM Period_Record
7 WHERE DATEDIFF(IFNULL(end_date, CURDATE()), start_date) > 40;
8
9
```

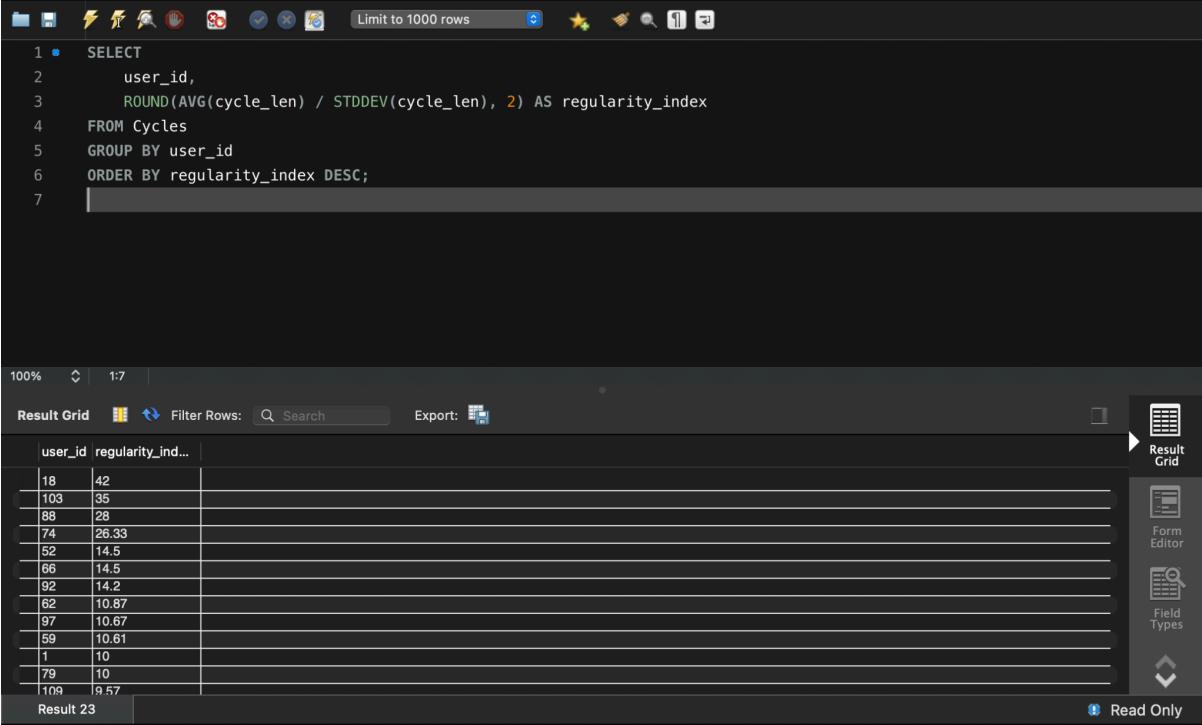
The bottom pane shows the results of the query in a grid format:

user_id	record_id	start_date	end_date
3	8	2025-03-12	NULL
4	11	2025-01-21	NULL
5	14	2025-03-02	NULL
6	17	2025-04-17	NULL
8	23	2025-02-17	NULL
10	28	2025-03-06	NULL
11	29	2025-01-12	NULL
11	30	2025-02-12	NULL
12	31	2025-01-19	NULL
12	32	2025-02-17	NULL
13	34	2025-02-11	NULL
13	35	2025-04-07	NULL
14	38	2025-02-13	NULL

The results show 14 rows, indicating 14 missed or skipped cycles.

m. Cycle Regularity Index (Higher index = more regular cycles.)

```
SELECT
    user_id,
    ROUND(AVG(cycle_len) / STDDEV(cycle_len), 2) AS regularity_index
FROM Cycles
GROUP BY user_id
ORDER BY regularity_index DESC;
```



The screenshot shows a database query interface with the following details:

- Query Editor:** Displays the SQL code with line numbers 1 through 7.
- Result Grid:** Shows the output of the query in a tabular format. The columns are labeled "user\_id" and "regularity\_index". The data rows are:

user_id	regularity_index
18	42
103	35
88	28
74	26.33
52	14.5
66	14.5
92	14.2
62	10.87
97	10.67
59	10.61
1	10
79	10
109	9.57

- Toolbar:** Includes icons for file operations, search, and export.
- Right Panel:** Contains icons for "Result Grid", "Form Editor", and "Field Types".
- Status Bar:** Shows "Result 23" and "Read Only".

n. Find users with the highest number of Heavy flow days

```
SELECT
    u.user_id,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    COUNT(*) AS heavy_days
FROM Flow f
JOIN User u ON f.user_id = u.user_id
WHERE f.flow_type = 'Heavy'
GROUP BY u.user_id
ORDER BY heavy_days DESC
LIMIT 5;
```

The screenshot shows a database query editor interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL query is displayed in a code editor window. The code is identical to the one provided above. In the bottom half of the interface, the results of the query are shown in a grid. The grid has three columns: 'user\_id', 'user\_name', and 'heavy\_days'. The data is as follows:

user_id	user_name	heavy_days
5	Neha Pillai	2
8	Divya Iyer	2
12	Ananya Rao	2
19	Charvi Shetty	2
40	Trisha Malhotra	2

Below the grid, it says 'Result 5'. To the right of the grid, there is a sidebar with three buttons: 'Result Grid' (which is selected), 'Form Editor', and 'Field Types'. At the bottom right of the interface, there is a 'Read Only' button.

o. Find users who most often log “Spotting” days

```
SELECT
    u.user_id,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    COUNT(*) AS spotting_days
FROM Flow f
JOIN User u ON f.user_id = u.user_id
WHERE f.flow_type = 'Spotting'
GROUP BY u.user_id
ORDER BY spotting_days DESC;
```

The screenshot shows a database query editor interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL query is displayed in a code editor with line numbers from 1 to 10. The results are shown in a grid below the query. The grid has three columns: user\_id, user\_name, and spotting\_da... (truncated). The data is as follows:

user_id	user_name	spotting_da...
15	Aarohi Bhattacharya	2
46	Gauri Rajput	1
83	Kashish Sood	1
105	Neelam Rathore	1
5	Neha Pillai	1
31	Pallavi Kulkarni	1
48	Bhavika Lal	1
65	Manvi Khatri	1
77	Preeti Rawat	1
103	Kriti Sood	1

On the right side of the interface, there is a sidebar with icons for Result Grid, Form Editor, and Field Types. The Result Grid icon is highlighted. At the bottom right, there is a "Read Only" button.

**p. Join Period\_Reminder and Phase\_Reminder to show all active reminders**

```
SELECT
    u.username,
    r.reminder_type,
    CASE
        WHEN r.reminder_type = 'Period' THEN CONCAT('Notify ', prr.days_before_predicted, ' days before period')
        WHEN r.reminder_type = 'Phase' THEN CONCAT('Notify at start of ', phr.phase, ' phase')
    END AS reminder_details
FROM Reminder r
LEFT JOIN Period_Reminder prr ON r.reminder_id = prr.reminder_id
LEFT JOIN Phase_Reminder phr ON r.reminder_id = phr.reminder_id
JOIN User u ON r.user_id = u.user_id
WHERE r.reminder_active = TRUE;
```

The screenshot shows a database query editor with the following details:

- Toolbar:** Includes icons for file operations, search, and export.
- Code Editor:** Displays the SQL query with line numbers 1 through 13.
- Result Grid:** Shows the output of the query with columns: username, reminder\_ty..., and reminder\_details. The data is as follows:

username	reminder_ty...	reminder_details
nehap	Phase	Notify at start of Follicular phase
nehap	Period	Notify 2 days before period
divyaiyer	Phase	Notify at start of Ovulation phase
ananyaraao	Period	Notify 3 days before period
charvis	Period	Notify 1 days before period
tavig	Phase	Notify at start of Luteal phase
pallavik	Phase	Notify at start of Menstrual phase
ritikar	Period	Notify 0 days before period
snehak	Phase	Notify at start of Menstrual phase
aditinal	Period	Notify 2 days before period
simrand	Phase	Notify at start of Luteal phase
mitalic	Phase	Notify at start of Ovulation phase
taishwarwai	Period	Notify 1 days before period

**Bottom Right Panel:** Includes buttons for Result Grid, Form Editor, and Field Types.