Here's a step-by-step approach:

Prerequisites:

MySQL: Already set up, and a user with the necessary permissions.

API Endpoint: `POST /api/orgs/invite` – This endpoint needs to accept player data and return the status of the invite.

Task Scheduler: Use `cron` (Linux) or another scheduler to run this every 5 minutes.

Solution Implementation:

Database Connection Setup:
Set up a connection to your MySQL database using a Python MySQL library like `mysql-connector-python`.

API Request Logic:
The script will send a `POST` request to the invite API and handle the response.

Update Database Logic:
Based on the response from the invite API, the script will update the player record in the database.

Scheduled Task:
Use `cron` to schedule the script to run every 5 minutes.

Code Example (Python):

```
import mysql.connector
import requests
import time

# Database connection settings
db_config = {
  'host': 'your_db_host',
  'user': 'your_db_user',
  'password': 'your_db_password',
  'database': 'your_db_name'
}

# API endpoint
API_URL = "https://your_api_url/api/orgs/invite"

# Starting Player ID (X) - you will manually define this
START_PLAYER_ID = 100  # Change this to the actual starting ID

# Establish a database connection
def connect_to_db():
  return mysql.connector.connect(
    host=db_config['host'],
    user=db_config['user'],
    password=db_config['password'],
    database=db_config['database']
  )
```

```
# Function to send invite via API and get the response
def send_invite(player_id, nickname):
    try:
        # Prepare data to send in API request
        payload = {
            'player_id': player_id,
            'nickname': nickname
        }
        response = requests.post(API_URL, data=payload)

        # Check if the response is successful
        if response.status_code == 200:
            return response.json()['status']
        else:
            return 'server_error'
    except Exception as e:
        return str(e)

# Function to process the players and invite them
def process_invites(start_id):
    # Connect to DB
    connection = connect_to_db()
    cursor = connection.cursor()

    # Query the next 20 players who are not invited
    cursor.execute("""
        SELECT id, nickname FROM players
        WHERE invited = 0 AND id >= %s
        ORDER BY id ASC
        LIMIT 20
    """, (start_id,))

    players = cursor.fetchall()

    if not players:
        print("No more players to invite.")
        return None  # Exit gracefully if no players are found

    # Loop through each player and send invite
    for player in players:
        player_id, nickname = player
        invite_status = send_invite(player_id, nickname)

        # Map the API response to the invite_comment
        if invite_status == 'invited_ok':
            invite_comment = 'invited_ok'
        elif invite_status == 'already_member':
            invite_comment = 'already_member'
        elif invite_status == 'already_invited':
            invite_comment = 'already_invited'
        elif invite_status == 'no_package':
            invite_comment = 'no_package'
        elif invite_status == 'not_found':
            invite_comment = 'not_found'
        else:
            invite_comment = 'server_error'
```

```
        # Update player status and invite_comment in the database
        cursor.execute("""
            UPDATE players
            SET invited = 1, invite_comment = %s
            WHERE id = %s
        """, (invite_comment, player_id))
        connection.commit()

    cursor.close()
    connection.close()

    # Return the ID of the last player processed, to be used for the next run
    return players[-1][0]

# Main loop to run every 5 minutes
def main():
    start_id = START_PLAYER_ID
    while True:
        start_id = process_invites(start_id)
        if start_id is None:
            break  # Exit if no more players to invite
        print(f"Waiting 5 minutes before next run...")
        time.sleep(300)  # Wait for 5 minutes before the next execution

if __name__ == "__main__":
    main()
```