Revision 02

Questions Sample Answers

Database System - 2025

Contents

How to?..... 1
Question Sets considered..... 1
Shobuj Sir..... 2
Tips..... 12
Murad Sir..... 13

How to?

Highlighted texts are something you have to work on your own, or my comments :)
LLM = Large language model, eg. deepseek, qwen, ChatGPT...

Question Sets considered

1. Session 20-21
2. Session 19-20
3. Session 18-19

Can be **inaccurate**! Feel free to **criticize**.

Shobuj Sir

Various terminology with properties of RDBMS

1. Schema
2. Instance
3. Entity
4. Attribute
5. Tuple
6. Relations
7. Keys
8. Data integrity
9. … *(use LLM)*

Relational algebra based on university schema

- classroom (building, room_number, capacity)
- department (dept_name, building, budget)
- course (course_id, title, dept_name, credits)
- instructor (ID, name, dept_name, salary)
- section (course_id, sec_id, semester, year, building, room_number, time_slot_id)
- teaches (ID, course_id, sec_id, semester, year)
- student (ID, name, dept_name, tot_cred)
- takes (ID, course_id, sec_id, semester, year, grade)
- advisor (s_ID, i_ID)
- time_slot (time_slot_id, day, start_time, end_time)
- prereq (course_id, prereq_id)

a. Find instructors from the "Physics" department:

$$\sigma_{\text{dept\_name} = \text{'Physics'}}(instructor)$$

b. Find the ID and name of each instructor in a department located in the building "Watson".

$$\Pi_{\text{ID, name}}(\sigma_{\text{building} = \text{'Watson'}}(instructor \not\bowtie_{\text{instructor.dept\_name = department.dept\_name}} department))$$

c. Find the ID and name of each student who has taken at least one course in the "Comp. Sci." department.

$$\Pi_{\text{ID, name}}\left(\sigma_{\text{dept\_name}='Comp.\ Sci.'}\left(\left(student \bowtie_{\text{student.ID = takes.ID}} takes\right)\right) \bowtie_{\text{takes.course\_id = course.course\_id}} course\right)$$

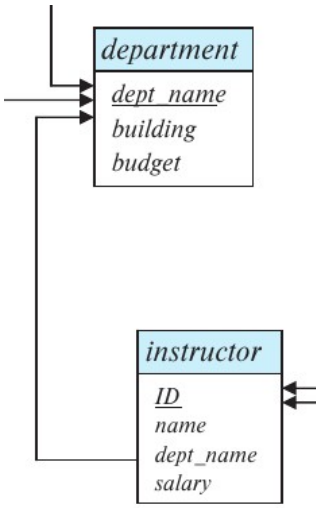
d. Find the ID and name of each student who has taken at least one course section in the year 2018.

$$\Pi_{\text{ID, name}}\left(\sigma_{\text{year}='2018'}\left(\left(student \bowtie_{\text{student.ID = takes.ID}} takes\right)\right) \bowtie_{\text{takes.course\_id = section.course\_id}} section\right)$$

e. Find the ID and name of each student who has not taken any course section in the year 2018.

$$\Pi_{\text{ID, name}}\left(\sigma_{\neg(\text{year}='2018')}\left(\left(student \bowtie_{\text{student.ID = takes.ID}} takes\right)\right) \bowtie_{\text{takes.course\_id = section.course\_id}} section\right)$$

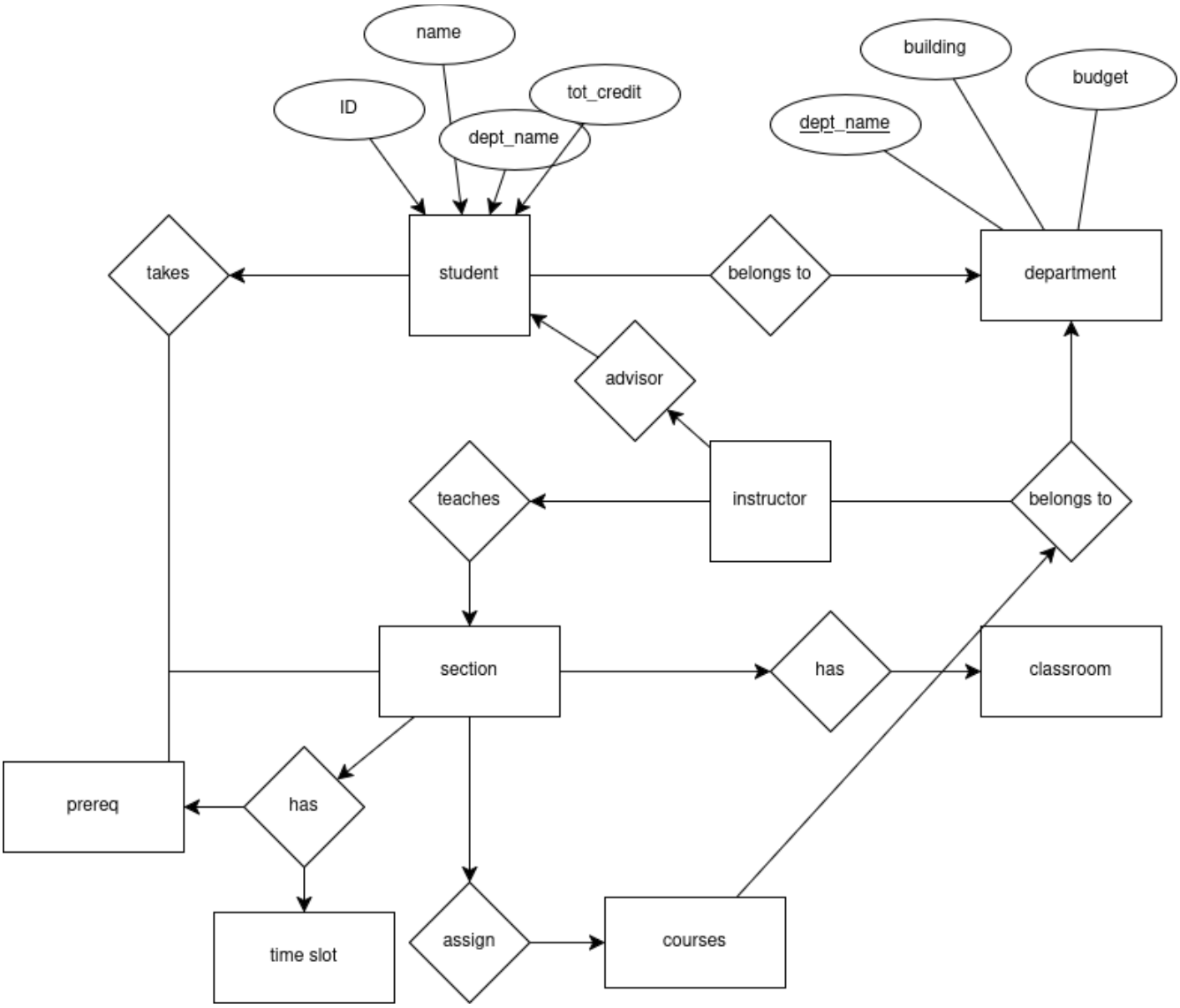
Consider the foreign-key constraint from the dept name attribute of instructor to the department relation. Give examples of inserts and deletes to these relations that can cause a violation of the foreign-key constraint.



If we try to insert the following data into the instructor relation we will break the integration,
(1024, “Sharafat”, “CSEES”, 0)
because the “CSEES” department doesn’t exist in the department relation.

And if we try to remove the following entry from the department relation, then it can break the integrity if CIT department has any instructor,
(“CIT”, “Library”, 10000)

ER diagram of batch management system



উপরের diagram আর university এর ERD একই রেখেছি। কাছাকাছি কনসেপ্ট, সামান্য ব্যতিক্রম = কিছু attribute কম থাকবে। পরীক্ষায় সবকিছূ মনে না থাকা স্বাভাবিক। যতোটুকু পারা যায়...

Suppose you are given a relation grade points(grade, points) that provides a conversion from letter grades in the takes relation to numeric scores. You may assume for simplicity that no takes tuple has the null value for grade.

Exercise এ সমাধান আছে, নিচের সমাধাণ ঠিক সেটারই কপি পেস্ট

a. Find the total grade points earned by the student with ID '12345', across all courses taken by the student.

```
select sum(credits * points)
from takes, course, grade_points
where takes.grade = grade_points.grade
and takes.course_id = course.course_id
and ID = '12345';
```

b. Find the grade point average (GPA) for the above student, that is, the total grade points divided by the total credits for the associated courses.

```
select
 sum(credits * points)/sum(credits) as GPA
from
 takes, course, grade_points
where
 takes.grade = grade_points.grade
and takes.course_id = course.course_id
and ID= '12345';
```

c. Find the ID and the grade-point average of each student.

```
select
 ID, sum(credits * points)/sum(credits) as GPA
from
 takes, ourse, grade_points
where
 takes.grade = grade_points.grade
and takes.course_id = course.course_id
group by ID;
```

d. Insert every student whose tot_cred attribute is greater than 100 as an instructor in the same department, with a salary of 10,000 taka.

```
insert into instructor
select ID, name, dept_name, 100000
from student
where tot_cred > 100;
```

The SQL like operator is case sensitive (in most systems), but the lower() function on strings can be used to perform case insensitive matching. To show how, write a query that finds departments whose names contain the string “sci” as a substring, regardless of the case.

```
select dept_name
```

```
from department
where lower(dept_name) like "%sci%";
```

Using the university schema, write an SQL query to find the IDs of those students who have retaken at least three distinct courses at least once (i.e., the student has taken the course at least two times).

```
SELECT t.ID
FROM takes t
WHERE t.grade IS NOT NULL
AND 2 <= (
     select count(course_id)
    from takes t2
    where t2.ID = t.ID
    and t2.course_id = t.course_id
)
GROUP BY t.ID
HAVING COUNT(DISTINCT t.course_id) >= 3;
```

SQL vs MySQL vs SQL Server

Aspect	SQL	MySQL	SQL Server
Type	Language	RDBMS (Database System)	RDBMS (Database System)
Developer	ISO / ANSI (standardized)	Originally MySQL AB, now Oracle	Microsoft
Purpose	Used to query and manage relational data	To store and manage data using SQL	To store and manage data using T-SQL
Language Dialect	SQL (standard)	SQL (with MySQL-specific extensions)	T-SQL (Transact-SQL, Microsoft's variant of SQL)
Open Source	N/A	Yes (community version)	No (proprietary, but has a free Express edition)
Platform Support	Runs inside DBMSs	Cross-platform (Linux, Windows, macOS)	Primarily Windows, some Linux support
Popular Use Case	Universal querying	Web development (LAMP stack)	Enterprise apps, BI, .NET integration
Performance	Depends on DBMS	Fast for read-heavy workloads	Optimized for large, complex enterprise workloads
Security	Depends on DBMS	Basic to moderate	Advanced, includes Active Directory integration
GUI Tools	Depends on DBMS	MySQL Workbench,	SQL Server Management

Aspect	SQL	MySQL	SQL Server
		phpMyAdmin	Studio (SSMS), Azure Studio
ACID Compliance	Depends on DBMS	Yes, mostly with InnoDB engine	Yes, fully ACID-compliant
Community Support	N/A	Large, open-source community	Strong, but more enterprise/corporate
Licensing	Open standard	GPL (free) or commercial (Oracle)	Commercial license (some free editions available)

9.12 Perhaps the most important data items in any database system are the passwords that control access to the database. Suggest a scheme for the secure storage of passwords. Be sure that your scheme allows the system to test passwords supplied by users who are attempting to log into the system.

Answer:

A scheme for storing passwords would be to encrypt each password (after adding randomly generated "salt" bits to prevent dictionary attacks), and then use a hash index on the user-id to store/access the encrypted password. The password being used in a login attempt is then encrypted (if randomly generated "salt" bits were used initially, these bits should be stored with the user-id and used when encrypting the user-supplied password). The encrypted value is then compared with the stored encrypted value of the correct password. An advantage of this scheme is that passwords are not stored in clear text, and the code for decryption need not even exist. Thus, "one-way" encryption functions, such as secure hashing functions, which do not support decryption can be used for this task. The secure hashing algorithm SHA-1 is widely used for such one-way encryption.

List all possible states through which a transaction may pass during its execution and explain why each state transition may occur.

Silber book – page 806

Explain the transaction property

ACID property with example, check book

[D] Consider this schedule of two transactions:

(a)

T1	T2
Read(X)	
	Read(X)
Write(Y)	
	Write(Y)
commit	
	commit

(b)

T1	T2
Read (A)	
	Write (B)
Write (A)	

Is this schedule: serializable? Conflict serializable? Or both explain your own answer?

Both of them are serializable + conflict serializable,

✅ *serializable = প্রথমে যদি T1 চালানো হয় এবং তারপর T2, serially, তবুও same result দিবে*

✅ *conflict serializable = গ্রাফে কোনো loop নাই, T1(write-Y) → T2(write-Y)*

List five responsibilities of a database-management system. For each responsibility, explain the problems that would arise if the responsibility were not discharged.

Some responsibility contains,

Redundancy minimize – without it, our database will use much more storage volume

Integrity maintain – without integrity, when one occurance of data will be updated, it won't update other occuraces automatically

Concurrency support – without concurrency if multiple users try to do same operation, system may not work in the expected way

Authentication – without it anyone can access or edit data which is out of the scope of his

Backup and restore – without it, we may lose data permanently

Check web or book for more responsibilities!

Assume that two students are trying to register for a course in which there is only one open seat. What component of a database system prevents both students from being given that last seat?

Transaction management is responsible for handling this case. This part of the database deals with consistency.

(Read main book for learning more)

File approach vs database approach and how is it different from traditional file system

Use LLM, here's a difference table,

File processing	DBMS
No centralized control of data	Data is controlled in one or multiple instance by one system
Data redundancy can be occurred by different files	Data redundancy is used to link multiple occurrence of same database
Change in one table doesn't change other same fields	Change in one occurrence applies in all other occurrences
Doesn't provide security, because authentication can't be applied	Provide authentication, so everyone can't access everything

List at least two reasons why database systems support data manipulation using a declarative query language such as SQL, instead of just providing a library of C or C++ functions to carry out data manipulation.

Firstly to make DBMS more easy and simple, SQL is used. C or C++ is much harder to write manually, and different people can use different types of code for doing the same thing, but their complexity will differ which will make it difficult to work on the same codebase.

Secondly, SQL can provide a standardization, where all of its code is highly optimized. Otherwise, programming languages could be implemented in different ways which will lack a default standard and optimization.

Consider the following relational database:
employee(e-name, street, city)
works(e-name, c-name, salary)
company(c-name, city)
manages(e-name, m-name)
For each of the following queries, give an expression in the relational algebra,

i) Find the names, street address, and cities of all employees who work for Rupali Bank and earn more than 50,000 taka per month. Assume each person works for at most one company.

ii) Find the names of all employees in this database who live in the same city as the company for which they work.

iii) Find the names of all employees who live in the same city and on the same street as do their managers.

iv) Find the names of all employees in this database who do not work for the First Bank Corporation. Assume that all people work for exactly one company.

Ans,

- **Query (i)** :

$$\pi_{[\text{e}-name,\,\text{street},\,\text{city}]}\left(\sigma_{[\text{c}-name=\text{Rupali}\,Bank\,\wedge\,\text{salary}>50000]}\left(\text{works}\bowtie_{\text{e}-\text{name}}\text{employee}\right)\right)$$

- **Query (ii)** :

$$\pi_{e-name}\left(\sigma_{employee.city=company.city}\left(\left(works\bowtie_{c-name}company\right)\bowtie_{e-name}employee\right)\right)$$

- **Query (iii)** :

$$\left(\sigma_{employee.street=employee2.street\,\wedge\,employee.city=employee2.city}\left(\left(\pi_{e-name}\left(manages\bowtie_{e-name}employee\right)\right)\bowtie_{m-name}\rho_{employee2}\left(employee\right)\right)\right)$$

- **Query (iv)** :

$$\pi_{e-name}\left(\sigma_{c-name\neq'FirstBankCorporation'}\left(works\right)\right)$$

[B.] Consider the employee database. Give an expression in the relational algebra to express each of the following queries: employee (person_name, street, city) works (person_name, company_name, salary) company (company_name, city) 4

a. Find the ID and name of each employee who works for "BigBank".
b. Find the ID, name, and city of residence of each employee who works for "BigBank".
c. Find the ID, name, street address, and city of residence of each employee who works for "BigBank" and earns more than $10000.
d. Find the ID and name of each employee in this database who lives in the same city as the company for which she or he works.

check exercise of chapter 2

Why NULL values might be introduced into a database

Check exercise of chapter 2

Where vs having clause

Where is used for setting a general condition all over the tuples, where having clause is used for aggregate functions.
Where is used before group by and having is used after group by.

3V of bigdata

check chapter 1 slide

SQL injection

chapter 9, section 9.8.1

"A schedule is called conflict serializable if after swapping of non-conflicting operations, it can transform into a serial schedule."

Check book (page 814) or LLM.

Tips

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

Figure 2.8 Schema of the university database.

Try to remember the **university schema**, as it may not be included in the question!
পরীক্ষায় এই ডায়াগ্রাম টা দেয়া নাও থাকতে পারে। এটা মনে রাখলে, ERD/ schema diagram/ SQL query/ algebra সবক্ষেত্রেই সুবিধা পাওয়া যাবে

Murad Sir

CSE Club of PSTU wants to give scholarship to some students on the following criteria:

a. Students must be female
b. Student do not get any other private scholarships such like DBBL (Dutch-bangla bank Ltd.) scholarship
c. Grade must be at least 3.25
d. Student should not be punished for any awful activity.

Create necessary table (yourself) and write necessary query for i, ii, iii and iv.

[A.] UGC wants to give scholarship to some students on the following criteria:

a. Student must be of CSE Faculty of PSTU (02 in the student ID means CSE Students)
b. Students must be female
c. Student do not get any other private scholarships such like **Ankur** scholarship
d. Grade must be at least 3.75
e. Student should not be punished for any awful activity

Create necessary table (yourself) and write necessary query for i, ii, iii, iv and v.

Table creation for CSE Club PSTU

For oracle

```
CREATE TABLE Students (
  student_id VARCHAR2(7) PRIMARY KEY,
  name VARCHAR2(100),
  gender VARCHAR2(10),
  grade NUMERIC(1, 3),
  private_scholarship VARCHAR2(3), -- 'Yes' or 'No'
  punished VARCHAR2(3) -- 'Yes' or 'No'
);
```

Data Insertion

```
INSERT INTO Students (student_id, name, gender, grade, private_scholarship, punished)
VALUES (2102024, 'sharafat', 'Male', 0.50, 'No', 'No');
```

Queries

Student of CSE

Here, in the ID, third and fourth digit's "02" means CSE student. How can I filter it according to this condition?

```
SELECT *
FROM Students
WHERE SUBSTR(student_id, 3, 2) = '02';
```

Students must be female

```
SELECT *
FROM Students
WHERE gender = 'Female';
```

Students don’t get private scholarship

```
SELECT *
FROM Students
WHERE private_scholarship = 'No';
```

Grade must be at least 3.25

```
SELECT *
FROM Students
WHERE grade >= 3.25;
```

Should not be punished for any awful activity

```
SELECT *
FROM Students
WHERE punished = 'No';
```

Give an example which shows a statement-level BEFORE DELETE trigger on the BOOKSHELF table. When a user attempts to delete a record from the BOOKSHELF table, this trigger is executed and checks two system conditions: that the day of the week is neither Friday nor Saturday, and that the Oracle usemame (Student ID) of the account performing the delete include the Student ID's 3rd and 4th digit equal "02" in respect of PSTU ID management of the students.

```
CREATE OR REPLACE TRIGGER BOOKSHELF_BEF_DEL
BEFORE DELETE ON BOOKSHELF

DECLARE
  weekend_error EXCEPTION;
  not_library_user EXCEPTION;

BEGIN
  IF TO_CHAR(SYSDATE, 'DY') IN ('FRI', 'SAT') THEN
    RAISE weekend_error;
  END IF;

  IF SUBSTR(USER, 3, 2) != '02' THEN
    RAISE not_library_user;
  END IF;
EXCEPTION
```

```
  WHEN weekend_error THEN
    RAISE_APPLICATION_ERROR(-20001, 'Deletions not allowed on weekends');

  WHEN not_library_user THEN
    RAISE_APPLICATION_ERROR(-20002, 'Deletions only allowed by Library users');
END;
/
```

উপরের query টা আমাদের সিলেবাস বহির্ভূত – স্যার পড়ান নি

আর update এবং update+ insert শর্তের উপর trigger এর উদাহারণ বই এ দেয়া আছে, সেটা দেখলেই হবে ↓

শুধুমাত্র update শর্তের উপর কাজ করার trigger:

```
create or replace trigger BOOKSHELF_BEF_UPD_ROW
before update on BOOKSHELF
for each row
when (new.Rating < old.Rating)
begin
    insert into BOOKSHELF_AUDIT
        (Title, Publisher, CategoryName,
        Old_Rating, New_Rating, Audit_Date)
    values
        (:old.Title, :old.Publisher, :old.CategoryName,
        :old.Rating, :new.Rating, Sysdate);
end;
/
```

যখন একই সাথে update এবং insert এর উপর trigger বসাতে হবে,

```
create or replace trigger BOOKSHELF_BEF_UPD_INS_ROW
before insert or update of Rating on BOOKSHELF
for each row
begin
  if INSERTING then
    insert into BOOKSHELF_AUDIT
      (Title, Publisher, CategoryName,
      New_Rating, Audit_Date)
    values
      (:new.Title, :new.Publisher, :new.CategoryName,
```

```
        :new.Rating, Sysdate);
  else
    insert into BOOKSHELF_AUDIT
      (Title, Publisher, CategoryName,
      Old_Rating, New_Rating, Audit_Date)
    values
      (:old.Title, :old.Publisher, :old.CategoryName,
      :old.Rating, :new.Rating, Sysdate);
  end if;
end;
/
```

Expired account vs locked account

Account expires due to PASSWORD_LIFE_TIME limit. Then we are asked to change the password for PASSWORD_GRACE_TIME. If we don't change password in that certain number of days, we are forced to change the password.
On the other hand, if FAILED_LOGIN_ATTEMPTS is set in the profile, then rewriting wrong password that amount of time will get the account locked. To unlock we have to wait certain days as defined in the PASSWORD_LOCK_TIME.

PASSWORD_REUSE_TIME **and** PASSWORD_REUSE_MAX

If one is set to any value, other than unlimited, then the other limit must have to be set to a value other than unlimited. If both are unlimited then it's the default behavior, which means there's no restriction.
If one is set to some value and other one is set to unlimited, then we will never be able to use the same password, used previously.

একটা যদি কিছু ভ্যালু থাকে, অপরটা অবশ্যই কোনো ভ্যালু হতে হবে, একটা ভ্যালু সেট করা আরেকটা unlimited থাকতে পারবে না। তাহলে কখনোই একই পাসওয়ার্ড পুনরায় ব্যবহার করা যাবে না।

আর যদি দুটোই unlimited থাকে তাহলে কোনো শর্ত প্রযোজ্য হবে না।

যদি আরো কোনো প্রশ্নের উত্তর যুক্ত করার প্রয়োজনীয়তা থাকে,

Feel free to reach me out.

সবাই সবার জন্য দো'আ'র দরখাস্ত রইলো!