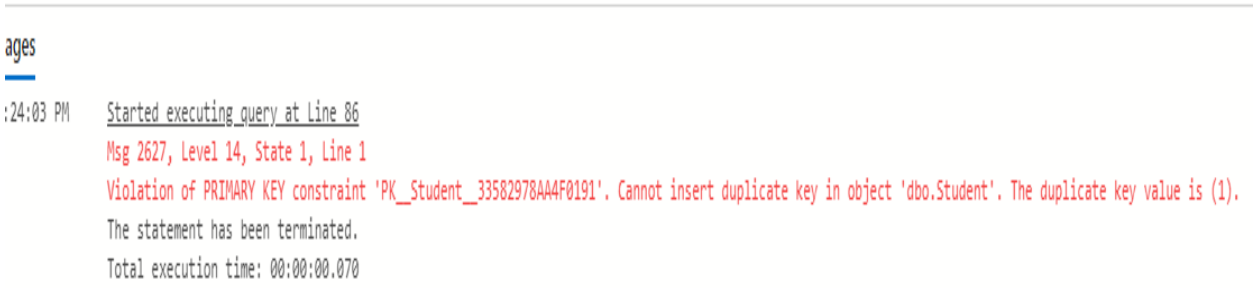
Problem 1:

1.1)

We have executed the following queries to check whether Azure SQL can check for violation of constraints. We have used the same database created as a part of the assignment 2.

1. Query executed to check for uniqueness

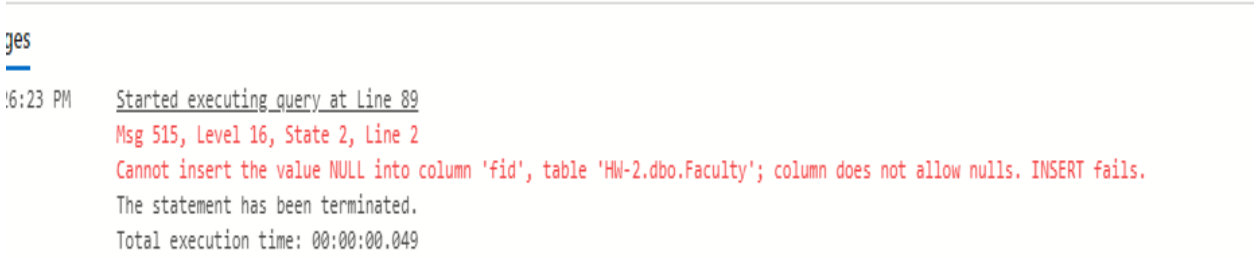
```
------Problem 1
--a)Uniqueness
INSERT INTO Student VALUES
(1,'VK','History','FR',18)
```



Azure SQL throwed the warning shown in the picture thus maintaining the uniqueness.

2. Query to check Not Null constraint

```
--b)Not null
INSERT INTO Faculty VALUES
( NULL,'Pearson',10,55000)
```



We have tried to insert a null value and Azure SQL didn't let that happen, Thereby, maintaining the Not Null constraint.

3. The following query has been executed to check for the insertion operation that violates foreign key constraint.

```
---c)Foriegn key violation
--Insert
INSERT INTO Enrolled VALUES
(12,'Data Networks')
```

```
:38 PM    Started executing query at Line 95
          Msg 547, Level 16, State 0, Line 1
          The INSERT statement conflicted with the FOREIGN KEY constraint "FK__Enrolled__snum__61BB7BD9". The conflict occurred in database "HW-2", table "dbo.Student", column 'snum'.
          The statement has been terminated.
          Total execution time: 00:00:00.120
```

Azure throws a conflict thus maintaining the foreign key constraint.

4. The following query has been executed to check for the deletion operation that violates foreign key constraint.

```
--DELETE
DELETE FROM Student WHERE snum=4
```

```
4:28 PM    Started executing query at Line 99
           Msg 547, Level 16, State 0, Line 1
           The DELETE statement conflicted with the REFERENCE constraint "FK__Enrolled__snum__61BB7BD9". The conflict occurred in database "HW-2", table "dbo.Enrolled", column 'snum'.
           The statement has been terminated.
           Total execution time: 00:00:00.054
```

Azure throws a conflict thus maintaining the foreign key constraint.

5. The following query has been executed to check for the update operation that violates foreign key constraint.

```
---UPDATE
UPDATE Student SET snum=30 WHERE snum=4
```

```
jes

4:28 PM    Started executing query at Line 99
           Msg 547, Level 16, State 0, Line 1
           The DELETE statement conflicted with the REFERENCE constraint "FK__Enrolled__snum__61BB7BD9". The conflict occurred in database "HW-2", table "dbo.Enrolled", column 'snum'.
           The statement has been terminated.
           Total execution time: 00:00:00.054
```

Azure throws a conflict thus maintaining the foreign key constraint.

6. The following query has been executed to check for the retrieval operation that violates foreign key constraint.

```
--d) Retrieval
SELECT sname FROM Student WHERE sname=4
```

```
Messages

:44 PM    Started executing query at Line 105
          Msg 245, Level 16, State 1, Line 1
          Conversion failed when converting the varchar value 'Adams' to data type int.
          Total execution time: 00:00:00.055
```

Azure throws a conflict thus maintaining the foreign key constraint.

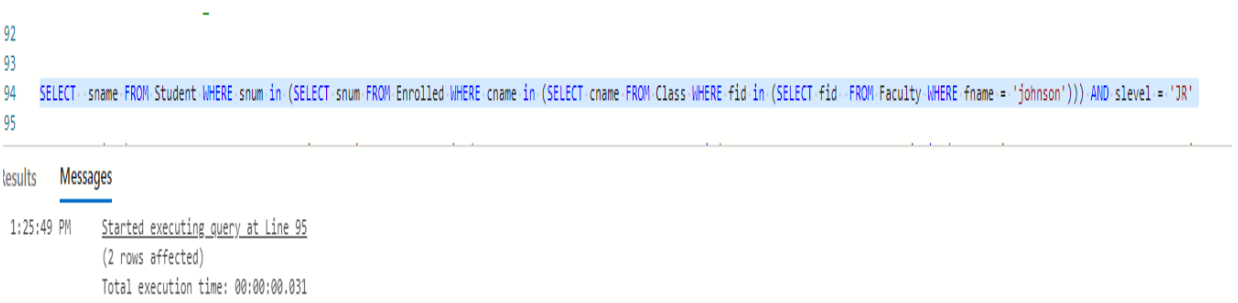
1.2)

We have chosen Student table to create index on. In specific we have chosen slevel form Student table to create index. It is a secondary key index because slevel is a non-candidate key. This is choice is made particularly as there are 4 queries concerning this attribute and azure already creates indexes on primary keys, explicitly creating indexing on this attribute would increase the query execution speed.

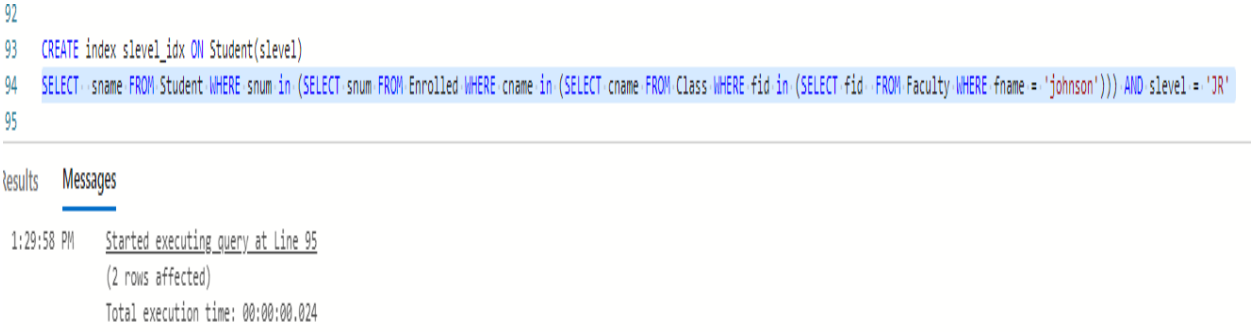
1. The following query should be executed to create an index

```
-- create index on slevel of students
CREATE index slevel_idx ON Student(slevel)
```

2. The following steps show the execution time for the queries run before and after the index creation.
   Query 1-



Query 1 after indexing



Observation- This indexing resulted in decrease in the execution time. The execution time changed from 00:00:00.031 to 00:00:00.024

Query2 –

```
100
101    SELECT slevel ,AVG(age) as average_age FROM Student GROUP BY slevel
102
103
```

Results Messages

```
1:33:41 PM     Started executing query at Line 102
               (3 rows affected)
               Total execution time: 00:00:00.030
```

Query 2 after indexing

```
100    CREATE index slevel_idx ON Student(slevel)
101    SELECT slevel ,AVG(age) as average_age FROM Student GROUP BY slevel
102
```

Results Messages

```
1:36:15 PM     Started executing query at Line 102
               (3 rows affected)
               Total execution time: 00:00:00.025
```

Observation- This indexing resulted in decrease in the execution time. The execution time changed from 00:00:00.030 to 00:00:00.025

Query-3

```
109
110    DELETE FROM Enrolled WHERE snum in (SELECT snum FROM Student WHERE slevel = 'SR' )
111
```

Messages

```
1:44:59 PM     Started executing query at Line 111
               (0 rows affected)
               Total execution time: 00:00:00.026
```

Query-3 after indexing

```
108    CREATE index slevel_idx ON Student(slevel)
109
110    DELETE FROM Enrolled WHERE snum in (SELECT snum FROM Student WHERE slevel = 'SR' )
111
112
```

lessages

```
1:47:16 PM     Started executing query at Line 111
               (0 rows affected)
               Total execution time: 00:00:00.021
```

Observation- This indexing resulted in decrease in the execution time. The execution time changed from 00:00:00.026 to 00:00:00.021

Query-4

```
113    DELETE FROM Student WHERE slevel = 'SR'
114
115
```

Messages

```
1:49:21 PM     Started executing query at Line 114
               (0 rows affected)
               Total execution time: 00:00:00.031
```

Query-4 after indexing

```
111
112    CREATE index slevel_idx ON Student(slevel)
113
114    DELETE FROM Student WHERE slevel = 'SR'
115
116
```

Messages

```
1:51:00 PM     Started executing query at Line 115
               (0 rows affected)
               Total execution time: 00:00:00.022
```

Observation- This indexing resulted in decrease in the execution time. The execution time changed from 00:00:00.031 to 00:00:00.022

The above information depicts that by indexing slevel we have reduced the execution time by minimizing the number of searches it does to find the information.

Problem 2)

Assumption:

For option=1: Department ID should already exist in the faculty table. If the value entered is other than the available deptId then it throws an error “Cannot insert the value NULL into column 'salary'”.

Stored procedure for Option-1

```
DROP PROCEDURE IF EXISTS insertintofaculty
GO
CREATE PROCEDURE insertintofaculty
@facid INT ,
@facname VARCHAR(64) ,
@facdept INT
AS
BEGIN
        DECLARE @av_salary FLOAT;
        SET @av_salary = (SELECT AVG(salary) FROM Faculty GROUP BY deptid HAVING deptid = @facdept);

        IF @av_salary > 50000

            INSERT INTO Faculty VALUES(@facid,@facname,@facdept,0.9*@av_salary);
        IF @av_salary < 30000

            INSERT INTO Faculty VALUES(@facid,@facname,@facdept,@av_salary);
        ELSE

            INSERT INTO Faculty VALUES(@facid,@facname,@facdept,0.8*@av_salary);
END
```

When option-3 is selected initially without any changes.

```
Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
```

i) After inserting one faculty using Option-1. Updated faculty table.1

```
4) Exit.
1
Please enter integer faculty ID:
105
Please enter faculty name:
Smith
Please enter department id:
12
Connecting to the database...


Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
105 | Smith | 12 | 24000.0
```

ii) After inserting 2nd faculty using Option-1. Updated faculty table.1

```
1
Please enter integer faculty ID:
106
Please enter faculty name:
James
Please enter department id:
11
Connecting to the database...


Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
105 | Smith | 12 | 24000.0
106 | James | 11 | 25600.0
```

iii) After inserting 3rd faculty using Option-1. Updated faculty table.1

```
Please enter integer faculty ID:
107
Please enter faculty name:
Steve
Please enter department id:
10
Connecting to the database...
```

```
Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
105 | Smith | 12 | 24000.0
106 | James | 11 | 25600.0
107 | Steve | 10 | 48000.0
```

Stored procedure for Option-2

```
DROP PROCEDURE IF EXISTS insertintofaculty2
GO
CREATE PROCEDURE insertintofaculty2
@facid2 INT,
@facname2 VARCHAR(64),
@facdeptid2 INT,
@exclude_deptid INT
AS
BEGIN
	DECLARE @av_salary2 FLOAT;
	SET @av_salary2 = (SELECT AVG(salary) FROM Faculty EXCEPT (SELECT salary FROM Faculty WHERE deptid = @exclude_deptid));

	INSERT INTO Faculty VALUES (@facid2 , @facname2, @facdeptid2, @av_salary2);

END
```

Before insertion the actual faculty table looks like below. With option-3 we can display the table.

```
Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
```

i) **After inserting one faculty using Option-2 with the following values.**

```
Please enter integer faculty ID:
105
Please enter faculty name:
James
Please enter department id:
10
Please enter exclude department id:
11
```

The Updated faculty table.

```
Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
105 | James | 10 | 50000.0
```

ii) **After inserting 2nd faculty member using Option-2 with the following values.**

```
Please enter integer faculty ID:
106
Please enter faculty name:
Peter
Please enter department id:
14
Please enter exclude department id:
11
```

The Updated faculty table.

```
Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
105 | James | 10 | 50000.0
106 | Peter | 14 | 50000.0
```

iii) After inserting 3rd faculty member using Option-2 with the following values.

```
Please enter integer faculty ID:
107
Please enter faculty name:
Toni
Please enter department id:
12
Please enter exclude department id:
10
```

The Updated faculty table.

```
Contents of the Faculty table:
fid | fname | deptid | salary
101 | Johnson | 10 | 55000.0
102 | Lynn | 10 | 65000.0
103 | Lynn | 12 | 30000.0
104 | Black | 11 | 32000.0
105 | James | 10 | 50000.0
106 | Peter | 14 | 50000.0
107 | Toni | 12 | 37333.33203125
```

Program termination with Option 4:

```
Please select one of the options below:
1) Insert new Faculty member:
2) Insert new Faculty member with exclude option
3) Display all faculty
4) Exit!
4
Exiting! Goodbuy!
```

Question 3)

Given the following relational database table:
Patients'(**ID**, name, symptom, days_in_hospital)
The following insertions are performed on the table Patients:
Insert record <20, Johnson, cough, 3>
Insert record <10, Black, fever, 5>
Insert record <30, Miller, fever, 10>
Insert record <70, Brown, fatigue, 2>
Insert record <60, Grant, headache, 4>
Insert record <50, Miller, nausea, 15>
Insert record <90, Brown, cough, 8 >

Assume each block in the Patients file can store up to 2 patient records. Do the following:

1. Assuming that Patients is organized as a heap file, show the contents of the file after the last insertion.

Since in Heap file, the data is stored in the order it is inserted. It is not sorted on any of the search key and Also it is given in the question that each block on disk can only store two records. So the organization looks as follows.

ID	name	symptom	days_in_hospital	Block_number	Record address
20	Johnson	cough	3	0	1
10	Black	fever	5	0	2
30	Miller	fever	10	1	3
70	Brown	fatigue	2	1	4
60	Grant	headache	4	2	5
50	Miller	nausea	15	2	6
90	Brown	cough	8	3	7

2.Assuming that Patients is organized as a sequential file with days_in_hospital as the ordering field, show the contents (i.e. the data values as well as the associated block/bucket/record addresses) of the file after the last insertion.

Main data file

ID	name	symptom	days_in_hospital	Block_number	Record address
70	Brown	fatigue	2	0	1
20	Johnson	cough	3	0	2
60	Grant	headache	4	1	3
10	Black	fever	5	1	4
90	Brown	cough	8	2	5
30	Miller	fever	10	2	6
50	Miller	nausea	15	3	7

3.Assuming that Patients is organized as an index-sequential file on the search key days_in_hospital and assuming that the primary index, the secondary index on ID, and the secondary index on name have been, show the contents of Patients, the primary index, and the two secondary indices after the last insertion.

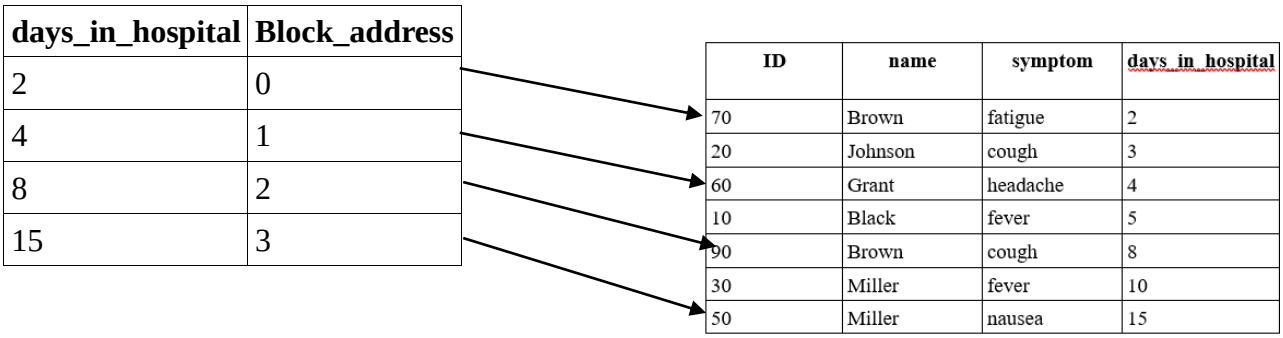
Main data file

ID	name	symptom	days_in_hospital	Block_number	Record address
70	Brown	fatigue	2	0	1
20	Johnson	cough	3	0	2
60	Grant	headache	4	1	3
10	Black	fever	5	1	4
90	Brown	cough	8	2	5
30	Miller	fever	10	2	6
50	Miller	nausea	15	3	7

Primary Index File (on days_in_hospital)

days_in_hospital	Block_address
2	0
4	1
8	2
15	3

The below representation can be assumed as how our primary key created will be visualized by the main memory.



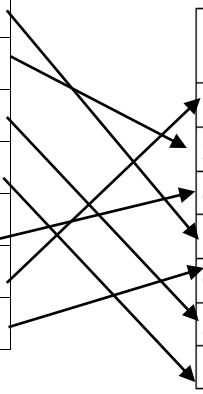
days_in_hospital	Block_address
2	0
4	1
8	2
15	3

ID	name	symptom	days_in_hospital
70	Brown	fatigue	2
20	Johnson	cough	3
60	Grant	headache	4
10	Black	fever	5
90	Brown	cough	8
30	Miller	fever	10
50	Miller	nausea	15

Secondary Index File on ID (Since ID is a primary key, it is a candidate key)

ID	Block_address
10	1
20	0
30	2
50	3
60	1
70	0
90	2

ID	Block_A
10	1
20	0
30	2
50	3
60	1
70	0
90	2



ID	name	symptom	days_in_hospital	Block_number
70	Brown	fatigue	2	0
20	Johnson	cough	3	0
60	Grant	headache	4	1
10	Black	fever	5	1
90	Brown	cough	8	2
30	Miller	fever	10	2
50	Miller	nausea	15	3

The secondary index created is based on the primary key of the table therefore keys are unique and block address is used to for the indexing and the above picture shows the visualization of the indexing in the main memory.

Secondary Index File on name (which is not a candidate key)

let the record address of the datafile be as follows

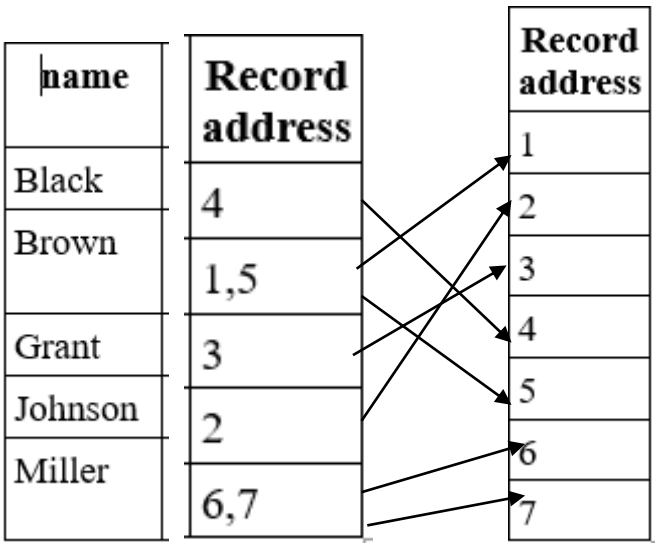
ID	name	symptom	days_in_hospital	Block_number	Record address
70	Brown	fatigue	2	0	1
20	Johnson	cough	3	0	2
60	Grant	headache	4	1	3
10	Black	fever	5	1	4
90	Brown	cough	8	2	5
30	Miller	fever	10	2	6
50	Miller	nausea	15	3	7

Bucket address to record address mapping (Linked list of record address)

Bucket_address	Record address
1000	4
1001	1,5
1002	3
1003	2
1004	6,7

Final mapping of the secondary index file on name.

name	Bucket_address	Record address
Black	1000	4
Brown	1001	1,5(linked list)
Grant	1002	3
Johnson	1003	2
Miller	1004	6,7(linked list)



The corresponding bucket data will be mapped to the respective records followed by the tupple that belongs to that particular record.

4. Given the index-sequential file organization as described in (3), explain step-by-step how the DBMS
would conduct search on this file organization to answer the following SQL query:

```
select name
    from Patients
        where ID between 30 and 60.
```

Ans)

Step -1)

First the DBMS would see the meta dictionary of the table to see if there is any index on the ID of the patients' table.

Step-2)

Now it notices that there is an index created on ID and then it checks with the buffer manager to see if the index file is in main memory. If yes, then it starts Step-3) else it fetches the index table from the disk and proceeds to step -3.

Step-3)

In this step the DBMS checks to see if the index table has the range 30 to 60 and gets the blocks 1,2,3 (3 I/O) from the disk and loads on to the main memory.

Step-4)

Now, the DBMS performs the sequential search on the blocks and retrieves the attribute name from the matched records and returns the names.