USE [Chess Game]

GO

SELECT COUNT(\*) FROM [RAW\_Chess\_Game] --20059 rows entered

--Looking at the input data into the RAW Table

SELECT \* FROM [RAW\_Chess\_Game]

--Dropping the table in case

IF OBJECT\_ID('[tmp\_Chess\_Game]') IS NOT NULL

DROP TABLE [tmp\_Chess\_Game]

--Since, the format of the colums is in Text, Creating a Working table with the proper column data types

--Creating a temporary table initially to check the normalised form (1NF) of this

CREATE TABLE tmp\_Chess\_Game(

id VARCHAR(100),

rated VARCHAR(10),

created\_at VARCHAR(100),

last\_move\_at VARCHAR(100),

turns INTEGER,

victory\_status VARCHAR(100),

winner VARCHAR(10),

increment\_code VARCHAR(100),

white\_id VARCHAR(100),

white\_rating INTEGER,

black\_id VARCHAR(100),

black\_rating INTEGER,

moves VARCHAR(2000),

opening\_eco VARCHAR(3),

opening\_name VARCHAR(100),

opening\_ply INTEGER

)

--Copying values from RAW to actual table

INSERT INTO tmp\_Chess\_Game(

id,

rated,

created\_at,

last\_move\_at,

turns,

victory\_status,

winner,

increment\_code,

white\_id,

white\_rating,

black\_id,

black\_rating,

moves,

opening\_eco,

opening\_name,

opening\_ply

)

SELECT id,

rated,

created\_at,

last\_move\_at,

turns,

victory\_status,

winner,

increment\_code,

white\_id,

white\_rating,

black\_id,

black\_rating,

moves,

opening\_eco,

opening\_name,

opening\_ply

FROM [RAW\_Chess\_Game]

--20059 rows affected

--Having a look at the temporary table

SELECT \* FROM [tmp\_Chess\_Game]

--INF INVESTIGATION

--CASE 1: Relational Schema has atomic cell value: Justified

--CASE 2: Dupliction of records

SELECT COUNT(\*) FROM [tmp\_Chess\_Game] --20059 rows

SELECT COUNT(\*) FROM (

SELECT DISTINCT \* FROM [tmp\_Chess\_Game]

) AS tmp --19630 distinct rows

--The above query displays the unique tuples in the table but there are some entries where the game id is the same but there is a mismatch of the case of the data entered.

--Hence, checking the unique Game ids

SELECT COUNT(DISTINCT [id]) FROM [tmp\_Chess\_Game] --19114 distinct game ids

--The difference in the rows denote the presence of duplicate value, hence removing all the duplicate entries into a new table

/\*

Checking the rows with same game id:

SELECT [id], COUNT([id]) FROM [tmp\_Chess\_Game]

GROUP BY [id]

HAVING COUNT([id]) = 1

ORDER BY COUNT([id]) DESC

SELECT \* FROM [tmp\_Chess\_Game]

WHERE [id] = '079kHDqh'

SELECT [id], COUNT([id]) FROM (

SELECT DISTINCT \* FROM [tmp\_Chess\_Game]

) AS tmp

GROUP BY [id]

HAVING COUNT([id]) >1

ORDER BY COUNT([id]) DESC

SELECT \* FROM (

SELECT DISTINCT \* FROM [tmp\_Chess\_Game]

) AS tmp

WHERE [id] = '079kHDqh'

SELECT [id], COUNT([created\_at]) FROM (

SELECT DISTINCT \* FROM [tmp\_Chess\_Game]

) AS tmp

WHERE [created\_at] NOT LIKE '%+%'

GROUP BY [id]

HAVING COUNT([id]) = 1

ORDER BY COUNT([id]) DESC

SELECT \* FROM [tmp\_Chess\_Game]

WHERE [id] = 'ZPHBiKBY'

\*/

--Dropping table Chess\_Game

IF OBJECT\_ID('[Chess\_Game]') IS NOT NULL

DROP TABLE [Chess\_Game]

--Creating a new Chess table from the non dupliacte values

SELECT \*

INTO [Chess\_Game]

FROM (

SELECT DISTINCT \* FROM [tmp\_Chess\_Game]

) AS tmp

WHERE [id] IN (

SELECT [id] FROM (

SELECT DISTINCT \* FROM [tmp\_Chess\_Game]

) AS tmp

GROUP BY [id]

HAVING COUNT([id]) = 1

)

OR

(

[id] IN (

SELECT [id] FROM (

SELECT DISTINCT \* FROM [tmp\_Chess\_Game]

) AS tmp

GROUP BY [id]

HAVING COUNT([id]) > 1

)

AND

[created\_at] NOT LIKE '%+%'

)

--19114 rows affected

/\*

Now just double checking the count of all game ids

SELECT COUNT(DISTINCT [id]) FROM [Chess\_Game]

--19114 rows

--Code Justified

\*/

--The first row in the table has missing values, hence dropping the row

DELETE FROM [Chess\_Game]

WHERE [turns] = 0

--1 row affected

--Now adding a row colun to the relational schema

ALTER TABLE [Chess\_Game]

ADD [RowNumber] int identity(1,1)

SELECT COUNT(\*) FROM [Chess\_Game] --19113 rows total

SELECT \* FROM [Chess\_Game]

--1NF JUSTIFIED

--2NF INVESTIGATION

--Case: Functional Dependencies

--Candidate keys in question:

--Checking Candidate Keys

--Candidate Key 1: RowNumber

SELECT COUNT(id) FROM [Chess\_Game] --19113 rows

SELECT COUNT(DISTINCT [id]) FROM [Chess\_Game] --19113 distinct Game ids

--Candidate key 2: Game id

--There are just 2 candidte keys, and each column is dependent on th whole of candidate keys, hence no functional dependency exists.

--2NF JUSTIFIED

--3NF Investigation

--1. opening\_ply is dependent on a combination of non-prime attributes i.e. opening\_eco & opening\_name

--Moving these three columns into a separate table

SELECT [opening\_eco],[opening\_name],[opening\_ply]

INTO tmp\_Game\_openings

FROM [Chess\_Game]

--19113 rows affected

SELECT \* FROM tmp\_Game\_openings

--Checking the normalized forms for the temporary table created

SELECT COUNT(\*) FROM tmp\_Game\_openings --19113 rows

SELECT COUNT(\*) FROM (

SELECT DISTINCT \* FROM tmp\_Game\_openings

) AS tmp --1545 distinct rows

SELECT DISTINCT \*

INTO [Game\_Openings]

FROM tmp\_Game\_openings

--1545 rows affected

SELECT \* FROM Game\_Openings

--Candidate key for the table created above is a combiaton of opening\_eco & opening\_name

--NORMALISATION JUSTIFIED

--Dropping the rows in main table Chess\_table

ALTER TABLE [Chess\_Game]

DROP COLUMN [opening\_ply]

--Dropping the temporary Game Openings Table

DROP TABLE tmp\_Game\_openings

--Dropping the temporary Chess Game Table

DROP TABLE tmp\_Chess\_Game

/\*

The final 2 tables created are:

Chess Game:

Candidate Key: RowNumber, id

Foreign Key: Combination of opening\_eco & opening\_name

SELECT \* FROM [Chess\_Game]

Game Openings:

Candidate Key: Combination of opening\_eco & opening\_name

SELECT \* FROM [Game\_Openings]

\*/