Formative 1 – Data Quality and Performance in Action

For this formative, I worked with a dataset containing 770 student records, which included demographic information, gaming habits, and academic performance. Before I could begin any meaningful analysis, I had to clean and prepare the data. I evaluated its quality based on five key dimensions: accuracy, completeness, consistency, timeliness, and uniqueness.

Part 1 Data Quality Assessment and Improvement

One of the first things I addressed was the 'Sex' column, which originally used numeric codes (0 and 1) to represent gender. These codes weren't very clear, so I replaced them with the actual labels "Male" and "Female" using a simple formula in Excel. This small change helped improve both readability and consistency in the dataset. (fig1)

Next, I focused on the 'Percentage' column, which had a lot of formatting issues. Some values had percent signs, commas, or even double decimal points (e.g., "75..5%"). To handle this, I created two formulas. The first flagged entries with incorrect formatting (fig2), and the second attempted to clean and convert the values into a consistent decimal format. For example, "75.5%" became 0.755 (fig3) Any entries that were too messy to fix automatically were flagged for manual review.

I then checked for missing values using a regular expression pattern to identify empty cells (fig4). Fortunately, there weren't any, so the dataset passed the completeness check. I also carried out several rounds of checks for duplicate records, and none were found, ensuring uniqueness. However, I wasn't able to fully assess accuracy or timeliness because the dataset lacked source information and timestamps.

Raw Data:

	A	B	C	D	E	F	G	H	I	J	K
1	Sex	School Code	Playing Years	Playing Often	Playing Hours	Playing Games	Parent Revenue	Father Education	Mother Education	Grade	percentage
2	0	1	1	2	1	1	4	4	5	77.5	7750,00%
3	1	1	1	3	1	1	1	3	3	83	8300,00%
4	0	1	0	0	0	0	1	3	3	80	8000,00%
5	0	1	3	5	1	1	2	2	3	45	4500,00%
6	1	1	1	1	2	1	1	3	4	85	8500,00%
7	0	1	1	5	1	1	1	2	2	80	8000,00%
8	0	1	1	2	2	1	2	3	3	55	5500,00%
9	0	1	1	5	2	1	2	3	3	80	8000,00%
10	1	1	2	1	1	1	3	3	5	60	6000,00%
11	0	1	2	5	2	1	1	2	4	88	8800,00%
12	1	1	4	1	1	1	2	4	2	80	8000,00%
13	0	1	0	0	0	0	2	4	4	45	4500,00%
14	1	1	3	3	2	1	2	5	5	90	9000,00%
15	1	1	4	1	2	1	2	5	5	74	7400,00%
16	1	1	3	5	5	1	2	4	4	95	9500,00%
17	1	1	2	4	3	1	1	3	3	50	5000,00%

Fig 1:

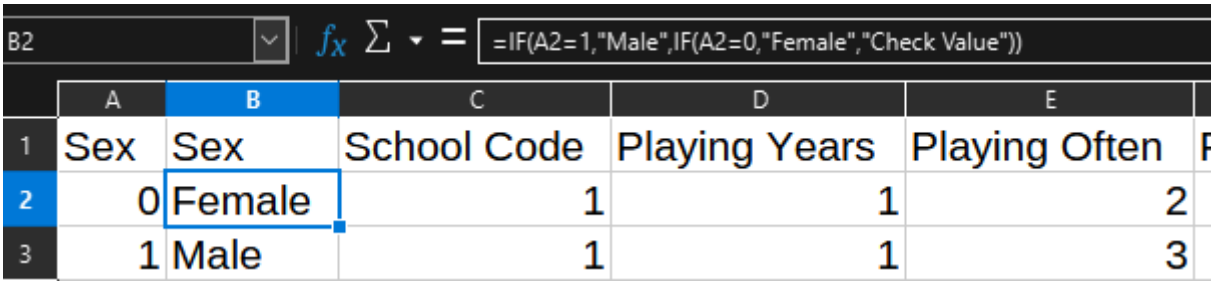
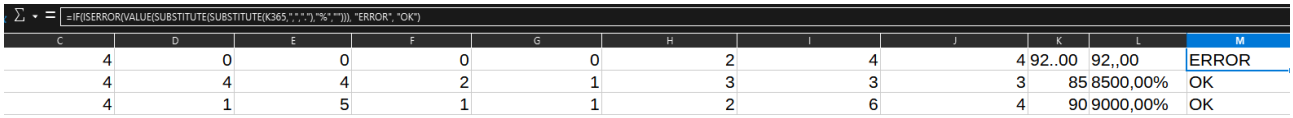
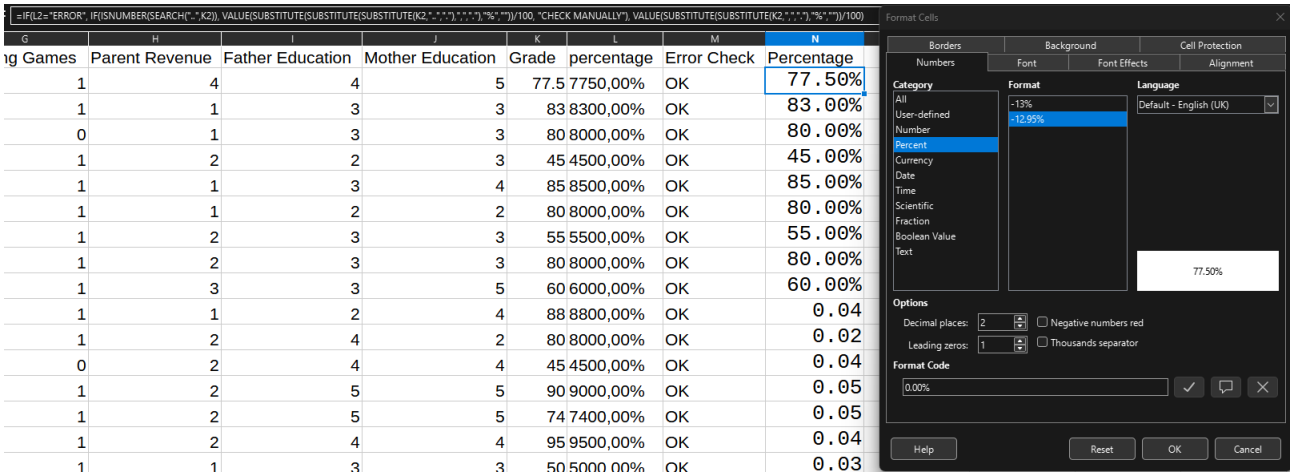


Fig 2:



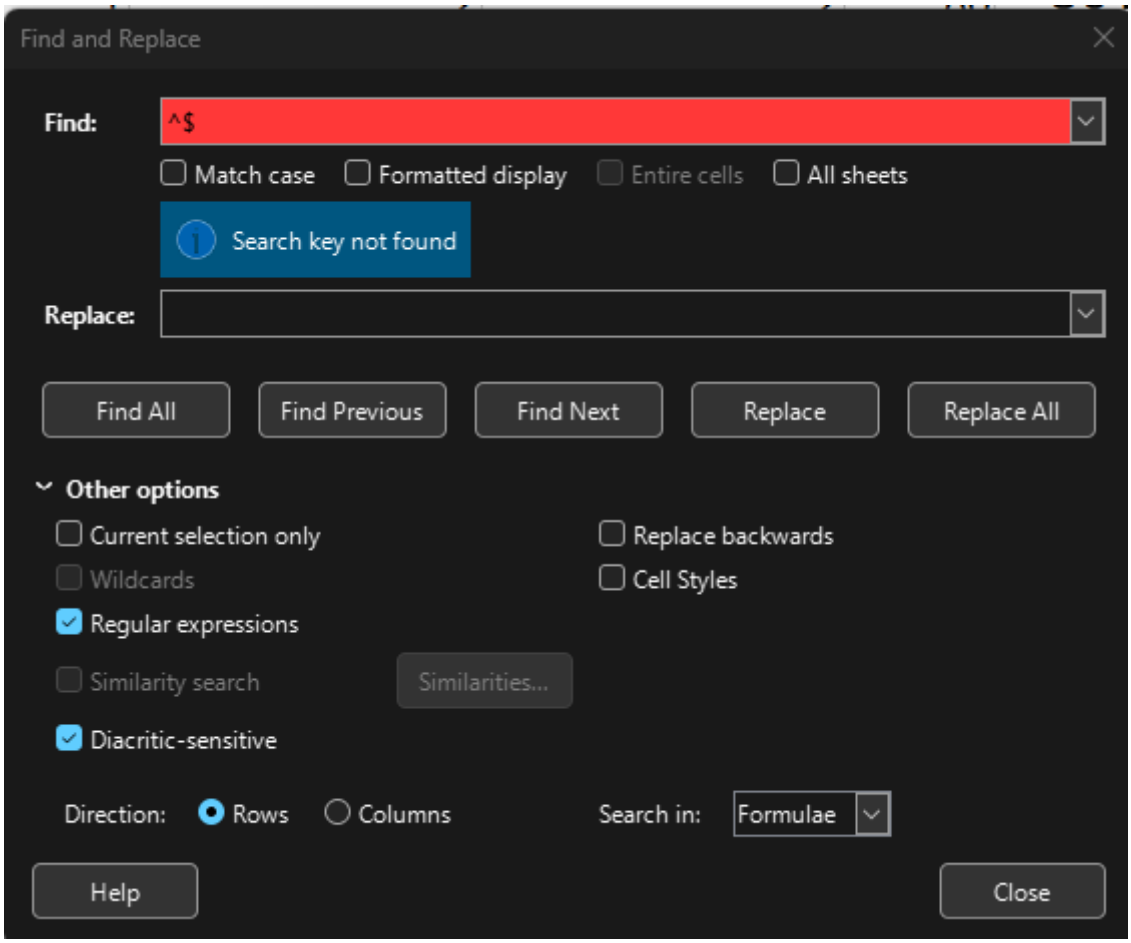
C	D	E	F	G	H	I	J	K	L	M
4	0	0	0	0	2	4	4	92	..00 92,,00	ERROR
4	4	4	2	1	3	3	3	85	8500,00%	OK
4	1	5	1	1	2	6	4	90	9000,00%	OK

Fig 3:



G	H	I	J	K	L	M	N
ıg Games	Parent Revenue	Father Education	Mother Education	Grade	percentage	Error Check	Percentage
1	4	4	5	77.5	7750,00%	OK	77.50%
1	1	3	3	83	8300,00%	OK	83.00%
0	1	3	3	80	8000,00%	OK	80.00%
1	2	2	3	45	4500,00%	OK	45.00%
1	1	3	4	85	8500,00%	OK	85.00%
1	1	2	2	80	8000,00%	OK	80.00%
1	2	3	3	55	5500,00%	OK	55.00%
1	2	3	3	80	8000,00%	OK	80.00%
1	3	3	5	60	6000,00%	OK	60.00%
1	1	2	4	88	8800,00%	OK	0.04
1	2	4	2	80	8000,00%	OK	0.02
0	2	4	4	45	4500,00%	OK	0.04
1	2	5	5	90	9000,00%	OK	0.05
1	2	5	5	74	7400,00%	OK	0.05
1	2	4	4	95	9500,00%	OK	0.04
1	1	3	3	50	5000,00%	OK	0.03

Fig 4:



Clean Data:

	B	C	D	E	F	G	H	I	J	K	N
1	Sex	School Code	Playing Years	Playing Often	Playing Hours	Playing Games	Parent Revenue	Father Education	Mother Education	Grade	Percentage
2	Female	1	1	2	1	1	4	4	5	77.5	77.50%
3	Male	1	1	3	1	1	1	3	3	83	83.00%
4	Female	1	0	0	0	0	1	3	3	80	80.00%
5	Female	1	3	5	1	1	2	2	3	45	45.00%
6	Male	1	1	1	2	1	1	3	4	85	85.00%
7	Female	1	1	5	1	1	1	2	2	80	80.00%
8	Female	1	1	2	2	1	2	3	3	55	55.00%
9	Female	1	1	5	2	1	2	3	3	80	80.00%
10	Male	1	2	1	1	1	3	3	5	60	60.00%
11	Female	1	2	5	2	1	1	2	4	88	88.00%
12	Male	1	4	1	1	1	2	4	2	80	80.00%
13	Female	1	0	0	0	0	2	4	4	45	45.00%

Part 2 Database Schema Design with SQL as a DDL

After cleaning the data, I designed a star schema (fig1) to make future analysis easier and more efficient. The central fact table (fact_grades, fig 2) stored the grades and an error flag to track any data issues. This table connected to four dimension tables:

- dim_student: included demographic data such as sex.
- dim_school: captured school identifiers.
- dim_parent_background: stored parental education (on a scale of 0–10) and income (scale of 1–4).
- dim_gaming_habits: recorded gaming behaviour, including how many years the student had played, how often, and for how long per session.

I enforced constraints to keep values within valid ranges, created indexes for faster joins, and enabled foreign key support in SQLite using PRAGMA foreign_keys = ON(fig3);. During the data load , I encountered a formatting error in one column, which I had to fix manually — a reminder that even with automation, human oversight is still important.

Example Use Case

Using the schema, I compared academic performance between frequent gamers (those with a playing frequency of 4 or more) and non-gamers (students with zero years of gaming). I filtered the results by high parental education levels to explore how socioeconomic factors might influence the outcomes.(fig4)

Conclusion

This project reinforced the importance of proper data cleaning and schema design. While there were some challenges — especially with formatting and missing metadata — the end result was a well-structured and reliable dataset that's ready for deeper analysis.

Fig 1:

```sql
-- Create dimension tables

-- This table stores information about the students.
CREATE TABLE dim_student (
    -- student_id: A unique identifier for each student. It's the primary key for this table and automatically increments.
    student_id INTEGER PRIMARY KEY AUTOINCREMENT,
    -- sex: The gender of the student (e.g., 'Male', 'Female'). It is a required field.
    sex VARCHAR(10) NOT NULL
);

-- This table stores information about the schools.
CREATE TABLE dim_school (
    -- school_id: A unique identifier for each school. It's the primary key and automatically increments.
    school_id INTEGER PRIMARY KEY AUTOINCREMENT,
    -- school_code: A unique numerical code assigned to each school. It is a required field.
    school_code INTEGER NOT NULL
);
```

```sql
s table stores information about the parents' background.
 TABLE dim_parent_background (
 parent_id: A unique identifier for each parent background record. It's the primary key and automatically increments.
rent_id INTEGER PRIMARY KEY AUTOINCREMENT,
 parent_revenue: A categorical representation of the parents' income level (0 to 4). It's a required field and has a check constraint to ensure values are within the valid range.
rent_revenue INTEGER NOT NULL CHECK (parent_revenue BETWEEN 0 AND 4),
 father_education: A categorical representation of the father's highest level of education (0 to 6). It's a required field with a check constraint for valid values.
ther_education INTEGER NOT NULL CHECK (father_education BETWEEN 0 AND 6),
 mother_education: A categorical representation of the mother's highest level of education (0 to 6). It's a required field with a check constraint for valid values.
ther_education INTEGER NOT NULL CHECK (mother_education BETWEEN 0 AND 6)
```

```sql
s table stores information about students' gaming habits.
 TABLE dim_gaming_habits (
 gaming_id: A unique identifier for each gaming habit record. It's the primary key and automatically increments.
ming_id INTEGER PRIMARY KEY AUTOINCREMENT,
 playing_years: The number of years the student has been playing games (0 to 4). It's a required field with a check constraint for valid values.
aying_years INTEGER NOT NULL CHECK (playing_years BETWEEN 0 AND 4),
 playing_often: A categorical representation of how often the student plays games (0 to 5). It's a required field with a check constraint for valid values.
aying_often INTEGER NOT NULL CHECK (playing_often BETWEEN 0 AND 5),
 playing_hours: A categorical representation of the number of hours the student spends playing games (0 to 5). It's a required field with a check constraint for valid values.
aying_hours INTEGER NOT NULL CHECK (playing_hours BETWEEN 0 AND 5),
 plays_games: A boolean value indicating whether the student plays games (TRUE or FALSE). It is a required field.
ays_games BOOLEAN NOT NULL
```

Fig 2:

<gameandgrade> enable foreign keys | <gameandgrade> Create Schema | *<gameandgrade> fact table create | <gameandgrade> Insert data to table | <gameandgrade> Test

```sql
Create fact table

This table stores the actual grades and related information, linking to the dimension tables.
ATE TABLE fact_grades (
 -- A unique identifier for each grade record. It's the primary key for this table and automatically increments.
 grade_id INTEGER PRIMARY KEY AUTOINCREMENT,
 -- A foreign key referencing the dim_student table, indicating which student achieved this grade. It is a required field.
 student_id INTEGER NOT NULL,
 -- A foreign key referencing the dim_school table, indicating which school the grade was achieved at. It is a required field.
 school_id INTEGER NOT NULL,
 -- A foreign key referencing the dim_parent_background table, linking the grade to the parent's background information. It is a required field.
 parent_id INTEGER NOT NULL,
 -- A foreign key referencing the dim_gaming_habits table, linking the grade to the student's gaming habits. It is a required field.
 gaming_id INTEGER NOT NULL,
 -- The actual grade achieved, stored as a decimal number with up to 5 total digits and 2 decimal places. It is a required field and has a check constraint
 grade DECIMAL(5,2) NOT NULL CHECK (grade),
 -- The grade represented as a percentage, stored as a decimal number with up to 5 total digits and 2 decimal places. It is a required field and has a check constraint.
 percentage DECIMAL(5,2) NOT NULL CHECK (percentage),
 -- An optional text field that can be used to flag any issues or anomalies related to this grade record.
 error_flag VARCHAR(10),
 -- FOREIGN KEY (student_id) REFERENCES dim_student(student_id): This establishes a foreign key relationship with the student_id column in the dim_student table.
 FOREIGN KEY (student_id) REFERENCES dim_student(student_id),
 -- FOREIGN KEY (school_id) REFERENCES dim_school(school_id): This establishes a foreign key relationship with the school_id column in the dim_school table.
 FOREIGN KEY (school_id) REFERENCES dim_school(school_id),
 -- FOREIGN KEY (parent_id) REFERENCES dim_parent_background(parent_id): This establishes a foreign key relationship with the parent_id column in the dim_parent_background table.
 FOREIGN KEY (parent_id) REFERENCES dim_parent_background(parent_id),
 -- FOREIGN KEY (gaming_id) REFERENCES dim_gaming_habits(gaming_id): This establishes a foreign key relationship with the gaming_id column in the dim_gaming_habits table.
 FOREIGN KEY (gaming_id) REFERENCES dim_gaming_habits(gaming_id)
```

Fig 3:

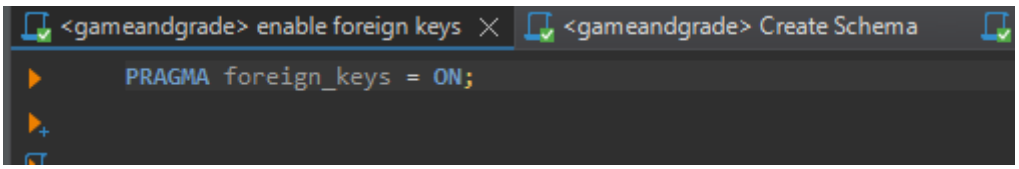
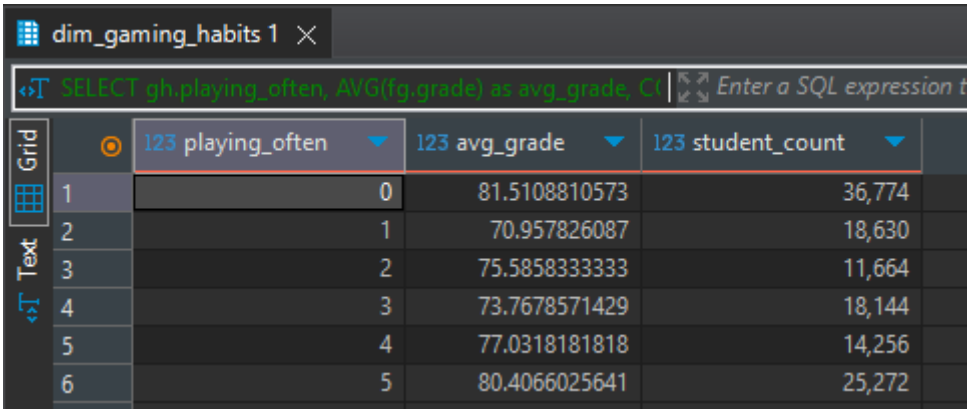


Fig 4:

```
grade by gaming frequency

ects the 'playing_often' category from the dim_gaming_habits table. This will be used to group the results.
ying_often,
culates the average of the 'grade' column from the fact_grades table for each gaming frequency group. The result is aliased as 'avg_grade'.
.grade) as avg_grade,
nts the number of students (grade records) within each gaming frequency group. The result is aliased as 'student_count'.
*) as student_count

cifies the fact_grades table (aliased as 'fg') as the primary table for this query.
rades fg

ns the fact_grades table with the dim_gaming_habits table (aliased as 'gh') using the common 'gaming_id' column. This links each grade record to the gaming habits of the student.
ming_habits gh ON fg.gaming_id = gh.gaming_id

ups the result set based on the 'playing_often' column from the dim_gaming_habits table. This allows the AVG and COUNT functions to operate on each distinct gaming frequency.
ying_often

ers the final result set by the 'playing_often' column in ascending order. This makes it easier to see the trend of average grades across different gaming frequencies.
ying_often;
```



	123 playing_often	123 avg_grade	123 student_count
1	0	81.5108810573	36,774
2	1	70.957826087	18,630
3	2	75.5858333333	11,664
4	3	73.7678571429	18,144
5	4	77.0318181818	14,256
6	5	80.4066025641	25,272