Formative 1- Data quality and performance in action

Part 1 – Data Quality Assessment and Improvement

This report evaluates a dataset of 770 student records (demographics, gaming habits, academic performance) against five core data quality dimensions: accuracy, completeness, consistency, timeliness, and uniqueness. The goal was to prepare the dataset for analysis by resolving structural and formatting issues.

Step 1: Sex Column Standardization

The Sex column initially used numeric codes (0 and 1). To improve interpretability, these were mapped to categorical values:

- Formula: =IF(A2=1,"Male",IF(A2=0,"Female","Check Value"))
- Outcome: Ensured consistency by replacing ambiguous codes with clear labels (Female/Male).

Step 2: Percentage Column Cleaning

The Percentage column had formatting inconsistencies (e.g., mixed symbols like "%", commas, double dots). Two formulas were applied:

1. Error Detection:
   =IF(ISERROR(VALUE(SUBSTITUTE(SUBSTITUTE(K2,",","."),"%",""))), "ERROR", "OK")
   Flagged cells with invalid formatting (e.g., "75..5%").
2. Normalization:
   =IF(L2="ERROR", IF(ISNUMBER(SEARCH("..",J2)),
   VALUE(SUBSTITUTE(SUBSTITUTE(SUBSTITUTE(J2,"..","."),",","."),"%",""))/100, "CHECK MANUALLY"), ...)
   Converted values to decimal percentages (e.g., "75.5%" → 0.755) and flagged ambiguous entries for manual review.

Step 3: Missing Value Checks

Using regex (^$), the dataset was scanned for empty cells. No missing values were identified, confirming completeness post-cleaning.

Limitations

- Accuracy/Timeliness: Could not be fully validated due to insufficient metadata (e.g., no source documentation or timestamps).
- Uniqueness: Duplicates were ruled out through iterative checks.

Outcome: The dataset is now consistent, with standardized formats and resolved errors, making it suitable for analysis.

Part 2 – Database Schema Design with SQL

A star schema was designed to analyze relationships between gaming habits and academic performance, prioritizing query efficiency.

Schema Structure

1. Fact Table: fact_grades

- Fields: grade, percentage, error_flag (validates data integrity).
- Foreign Keys: Links to all dimension tables.

2. Dimension Tables:
    - dim_student: Demographics (sex).
    - dim_school: School identifiers.
    - dim_parent_background: Parental education (0–10 scale) and income (1–4 scale).
    - dim_gaming_habits: Gaming metrics:
        - playing_years (0–4 years),
        - playing_often (0–5 frequency scale),
        - playing_hours (0–5 session length).

Key Design Features

- Constraints: Enforced valid ranges (e.g., parental education 0–10).
- Indexes: Optimized join performance between fact and dimension tables.
- Error Handling: An unresolved formatting error in column 365 was manually corrected post-load. This highlighted the need for stricter cleansing protocols.

SQLite Implementation

1. Foreign Key Enforcement: Enabled via PRAGMA foreign_keys = ON;.
2. Schema Creation: Tables built with explicit data types and constraints.
3. Data Transformation: Cleaned data loaded into the schema using batch inserts.

Example Analytical Use Case

Query: Compare academic performance between:

- Frequent gamers: playing_often ≥ 4 (high engagement).
- Non-gamers: playing_years = 0 (no engagement).

Filters: Parental education level (e.g., isolating students with parents scoring ≥8 on the education scale).

Outcome: The star schema allows efficient aggregation of grades across dimensions, enabling insights into how gaming habits correlate with academic results while controlling for socioeconomic factors.

Conclusion

This project underscores the importance of rigorous data cleaning and schema design in ensuring reliable analytics. While challenges like unresolved errors and metadata gaps persist, the refined dataset and optimized database structure provide a robust foundation for further analysis.

Screenshots

Part 1

Before:

	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%
18	0	1	1	5	1	1	1	4	1	98	9800,00%
19	1	1	2	2	3	1	2	5	3	90	9000,00%
20	1	1	3	3	2	1	3	3	4	87	8700,00%
21	0	1	0	0	0	0	2	2	3	55	5500,00%
22	0	1	2	2	2	1	4	4	4	95	9500,00%
23	1	1	1	4	1	1	1	3	2	70	7000,00%
24	1	1	1	5	1	1	3	4	1	70	7000,00%

During Clean:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Sex	School Code	Playing Years	Playing Often	Playing Hours	Playing Games	Parent Revenue	Father Education	Mother Education	Grade	percentage	Error Check	percentage
2	0	1	1	2	1	1	4	4	5	77.5	7750,00%	OK	77.50%
3	1	1	1	3	1	1	1	3	3	83	8300,00%	OK	83.00%
4	0	1	0	0	0	0	1	3	3	80	8000,00%	OK	80.00%
5	0	1	3	5	1	1	2	2	3	45	4500,00%	OK	45.00%
6	1	1	1	1	2	1	1	3	4	85	8500,00%	OK	85.00%
7	0	1	1	5	1	1	1	2	2	80	8000,00%	OK	80.00%
8	0	1	1	2	2	1	2	3	3	55	5500,00%	OK	55.00%
9	0	1	1	5	2	1	2	3	3	80	8000,00%	OK	80.00%
10	1	1	2	1	1	1	3	3	5	60	6000,00%	OK	60.00%
11	0	1	2	5	2	1	1	2	4	88	8800,00%	OK	88.00%
12	1	1	4	1	1	1	2	4	2	80	8000,00%	OK	80.00%
13	0	1	0	0	0	0	2	4	4	45	4500,00%	OK	45.00%
14	1	1	3	3	2	1	2	5	5	90	9000,00%	OK	90.00%
15	1	1	4	1	2	1	2	5	5	74	7400,00%	OK	74.00%
16	1	1	3	5	5	1	2	4	4	95	9500,00%	OK	95.00%
17	1	1	2	4	3	1	1	3	3	50	5000,00%	OK	50.00%
18	0	1	1	5	1	1	1	4	1	98	9800,00%	OK	98.00%
19	1	1	2	2	3	1	2	5	3	90	9000,00%	OK	90.00%
20	1	1	3	3	2	1	3	3	4	87	8700,00%	OK	87.00%
21	0	1	0	0	0	0	2	2	3	55	5500,00%	OK	55.00%
22	0	1	2	2	2	1	4	4	4	95	9500,00%	OK	95.00%
23	1	1	1	4	1	1	1	3	2	70	7000,00%	OK	70.00%
24	1	1	1	5	1	1	3	4	1	70	7000,00%	OK	70.00%
25	1	1	3	2	2	1	1	3	3	45	4500,00%	OK	45.00%
26	0	1	0	0	0	0	1	4	3	80	8000,00%	OK	80.00%
27	1	1	2	2	1	1	2	2	2	65	6500,00%	OK	65.00%
28	0	1	1	5	1	1	1	2	5	95	9500,00%	OK	95.00%
29	1	1	0	0	0	0	3	4	3	45	4500,00%	OK	45.00%
30	1	1	4	1	3	1	2	2	2	76.5	7650,00%	OK	76.50%

Error:

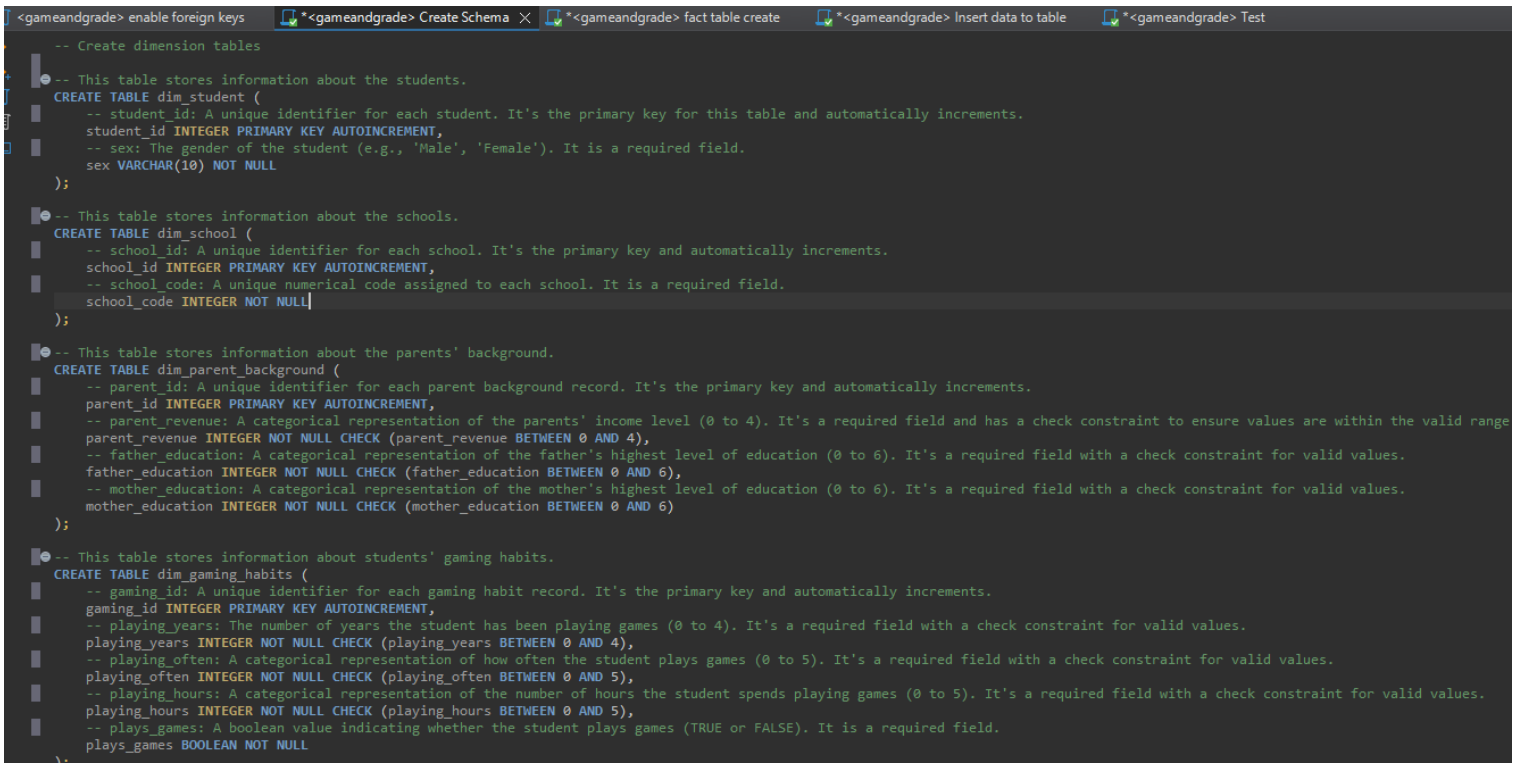
364	0	4	0	0	0	0	2	4	4	75	7500,00%	OK	75.00%
365	0	4	0	0	0	0	2	4	4	92..00	92,,00	ERROR	92.00%

Clean:

	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%
14	Male	1	3	3	2	1	2	5	5	90	90.00%
15	Male	1	4	1	2	1	2	5	5	74	74.00%
16	Male	1	3	5	5	1	2	4	4	95	95.00%
17	Male	1	2	4	3	1	1	3	3	50	50.00%
18	Female	1	1	5	1	1	1	4	1	98	98.00%
19	Male	1	2	2	3	1	2	5	3	90	90.00%
20	Male	1	3	3	2	1	3	3	4	87	87.00%
21	Female	1	0	0	0	0	2	2	3	55	55.00%
22	Female	1	2	2	2	1	4	4	4	95	95.00%
23	Male	1	1	4	1	1	1	3	2	70	70.00%
24	Male	1	1	5	1	1	3	4	1	70	70.00%
25	Male	1	3	2	2	1	1	3	3	45	45.00%
26	Female	1	0	0	0	0	1	4	3	80	80.00%
27	Male	1	2	2	1	1	2	2	2	65	65.00%
28	Female	1	1	5	1	1	1	2	5	95	95.00%

Part 2

Create Schema:



Create fact_table:

<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.
CREATE 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, ensuring data integrity.
    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, ensuring data integrity.
    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, ensuring data integrity.
    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, ensuring data integrity.
    FOREIGN KEY (gaming_id) REFERENCES dim_gaming_habits(gaming_id)
);
```

Load and transform data:

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

```sql
-- Insert distinct schools
INSERT INTO dim_school (school_code)
-- This INSERT statement populates the dim_school table with unique school codes.
SELECT DISTINCT "School Code"
-- It selects only the distinct values from the "School Code" column of the raw_import table.
FROM raw_import;

-- Insert distinct student demographics
INSERT INTO dim_student (sex)
-- This INSERT statement populates the dim_student table with unique student genders.
SELECT DISTINCT "Sex"
-- It selects only the distinct values from the "Sex" column of the raw_import table.
FROM raw_import;

-- Insert parent background data
INSERT INTO dim_parent_background (parent_revenue, father_education, mother_education)
-- This INSERT statement populates the dim_parent_background table with unique combinations of parent revenue, father's education, and mother's education.
SELECT DISTINCT "Parent Revenue", "Father Education", "Mother Education"
-- It selects only the distinct combinations of values from the specified columns of the raw_import table.
FROM raw_import;

-- Insert gaming habits
INSERT INTO dim_gaming_habits (playing_years, playing_often, playing_hours, plays_games)
-- This INSERT statement populates the dim_gaming_habits table with unique combinations of playing years, playing frequency, playing hours, and a boolean indicating if the student plays games.
SELECT DISTINCT
    "Playing Years",
    "Playing Often",
    "Playing Hours",
    -- This CASE statement derives the plays_games boolean value. If "Playing Years" is greater than 0, it's considered TRUE (1), otherwise FALSE (0).
    CASE WHEN "Playing Years" > 0 THEN 1 ELSE 0 END
FROM raw_import;

-- Insert fact data with joins to dimension tables
INSERT INTO fact_grades (
    student_id, school_id, parent_id, gaming_id,
    grade, percentage, error_flag
)
-- This INSERT statement populates the fact_grades table with grade information, linking records to the dimension tables.
SELECT
    -- This selects the student_id from the dim_student table by joining on the "Sex" column.
    s.student_id,
    -- This selects the school_id from the dim_school table by joining on the "School Code" column.
    sc.school_id,
    -- This selects the parent_id from the dim_parent_background table by joining on the "Parent Revenue", "Father Education", and "Mother Education" columns.
    p.parent_id,
    -- This selects the gaming_id from the dim_gaming_habits table by joining on the "Playing Years", "Playing Often", and "Playing Hours" columns.
    g.gaming_id,
    -- This selects the "Grade" directly from the raw_import table.
    r."Grade",
    -- This selects and cleans the "Percentage" from the raw_import table. It replaces commas with periods and removes the percentage sign, then casts it to a DECIMAL data type.
    CAST(REPLACE(REPLACE(r."Percentage", ',', '.'), '%', '') AS DECIMAL(5,2)),
    -- This selects the "Error Check" directly from the raw_import table.
    r."Error Check"
FROM
    -- This specifies the raw_import table as the primary source of data (aliased as 'r').
    raw_import r
    -- This joins raw_import with the dim_student table (aliased as 's') on the matching "Sex".
    JOIN dim_student s ON r."Sex" = s.sex
    -- This joins raw_import with the dim_school table (aliased as 'sc') on the matching "School Code".
    JOIN dim_school sc ON r."School Code" = sc.school_code
    -- This joins raw_import with the dim_parent_background table (aliased as 'p') on the matching "Parent Revenue", "Father Education", and "Mother Education".
    JOIN dim_parent_background p ON r."Parent Revenue" = p.parent_revenue
                                AND r."Father Education" = p.father_education
                                AND r."Mother Education" = p.mother_education
    -- This joins raw_import with the dim_gaming_habits table (aliased as 'g') on the matching "Playing Years", "Playing Often", and "Playing Hours".
    JOIN dim_gaming_habits g ON r."Playing Years" = g.playing_years
                            AND r."Playing Often" = g.playing_often
                            AND r."Playing Hours" = g.playing_hours;
```

Example Use Case