1-)

Erkin Alkan – 200709060

Baran İşci – 200709054

Sude Ebrar Çat – 200709007

→ Our aim is developing housewares database for the game called Animal Crossing on Nintendo Switch. This database includes the game items with their different properties.

2-) We changed lots of things in our database. We had a zero normalization form in our database in the start of the phase one but now we have three normalization form in our database. In the start we had a one big table in the middle and some columns distrubuted around it, we recognized some of the colums are not atomic and splitted them, then we used RDF method and foreign keys to establish relations between the tables and loaded all of our data.

3-) Yes, we managed to load all of our data to our database. We splitted our csv file columns. We prepared our EER Diagram to upload MySQL Workbench. We used Forward Engineering. We upload our database but there is no insert. We inserted our database with code below:

First we found our path to upload our csv files with this command;

show variables like "%secure%";

Then we used this command (example of loading hha2item table to database);

LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\hha2item.csv'

INTO TABLE hha2item

CHARACTER SET utf8

FIELDS TERMINATED BY ','

LINES TERMINATED BY '\n';

4-) We used MySQL Workbench in our project.

5-) Stored Procedures;

set @colorcount = 0;

call countColorsById('Beige',@colorcount) -> Includes in and out parameters, first parameter is a color name second parameter is color counter. This stored

procedure prints the how many color we had in our database in the given name of the parameter.

After calling stored procedure we need to print colorcount;

select @colorcount;

set @var = ‘Does not play music’;

call selectPropertiesBySpeakerType(@var) -> Includes inout parameter, given speaker type will be turned as ‘Retro’. It also selects all the speaker types with var before turning retro.

set @variation = ‘Black’;

call getVariation(variation) -> Includes in parameter, prints all items that has given parameter as variation.

Views;

Select * from baby_chairs -> prints out baby chairs with their properties.

6-) Our favorite stored procedure is countColorsById gives output as second parameter colorcount 451 for the color Beige.

set @colorcount = 0;

call countColorsById(‘Beige’,@colorcount);

select @colorcount;

prints 451.

7-) As the size of the database grows, we might experience performance issues, especially if indexes are not optimized. So we need to regularly analyse and optimize database indexes.

The database schema may not be fully normalized, leading to data redundancy and potential update anomalies. We may review the schema and normalize it to eliminate redundancy. Ensure that each piece of information is stored in only one place to maintain data integrity.

Complex queries might impact performance, readability, and maintenance. We need to break down complex queries into smaller, more manageable parts. We should optimize queries and use appropriate indexing.

8-)

CREATE TABLE source (

```
  `source_id` INT NOT NULL AUTO_INCREMENT, -- Sample: 1
  `source` VARCHAR(100) NOT NULL, -- Sample: Birthday
  `source_notes` TEXT NULL, -- Sample: Trade 3 insects to receive a model in the mail the next day
  PRIMARY KEY (`source_id`))
CREATE TABLE  color2item (
  `items_internal_id` INT NOT NULL, -- Sample: 383
  `items_variant_1` INT NOT NULL, -- Sample: 0
  `items_variant_2` INT NOT NULL, -- Sample: 0
  `color_color_id` INT NOT NULL, -- Sample: 5
  PRIMARY KEY (`items_internal_id`, `items_variant_1`, `items_variant_2`, `color_color_id`),
  INDEX `fk_item2color_items1_idx` (`items_internal_id` ASC, `items_variant_1` ASC, `items_variant_2` ASC) VISIBLE,
  CONSTRAINT `fk_item2color_color1`
    FOREIGN KEY (`color_color_id`)
  CONSTRAINT `fk_item2color_items1`
    FOREIGN KEY (`items_internal_id` , `items_variant_1` , `items_variant_2`)
    REFERENCES `mydb`.`items` (`internal_id` , `variant_1` , `variant_2`)

CREATE TABLE color (
  `color_id` INT NOT NULL AUTO_INCREMENT, -- Sample: 1
  `color_name` VARCHAR(45) NOT NULL,  -- Sample: Beige
  PRIMARY KEY (`color_id`),
  UNIQUE INDEX `color_name_UNIQUE` (`color_name` ASC) VISIBLE)

CREATE TABLE  numerics2item (
  `items_internal_id` INT NOT NULL, -- Sample: 383
```

```
  `items_variant_1` INT NOT NULL, -- Sample: 0
  `items_variant_2` INT NOT NULL, -- Sample: 0
  `numerics_numerics_id` INT NOT NULL, -- Sample: 2
  `numerics_value` FLOAT NULL, -- Sample: 3210.0
  PRIMARY KEY (`items_internal_id`, `items_variant_1`, `items_variant_2`, `numerics_numerics_id`),
  INDEX `fk_numerics2item_numerics1_idx` (`numerics_numerics_id` ASC) VISIBLE,
  CONSTRAINT `fk_numerics2item_items1`
    FOREIGN KEY (`items_internal_id` , `items_variant_1` , `items_variant_2`)
    REFERENCES `mydb`.`items` (`internal_id` , `variant_1` , `variant_2`)
  CONSTRAINT `fk_numerics2item_numerics1`
    FOREIGN KEY (`numerics_numerics_id`)
    REFERENCES `mydb`.`numerics` (`numerics_id`)

CREATE TABLE numerics (
  `numerics_id` INT NOT NULL AUTO_INCREMENT, -- Sample: 1
  `numerics_name` VARCHAR(45) NOT NULL, -- Sample: buy
  PRIMARY KEY (`numerics_id`))

CREATE TABLE  properties (
  `properties_id` INT NOT NULL AUTO_INCREMENT, -- Sample: 1
  `property_name` VARCHAR(100) NOT NULL, -- Sample: tag
  PRIMARY KEY (`properties_id`))

CREATE TABLE properties2item (
  `items_internal_id` INT NOT NULL, -- Sample: 383
  `items_variant_1` INT NOT NULL, -- Sample: 0
```

```
  `items_variant_2` INT NOT NULL, -- Sample: 0
  `properties_properties_id` INT NOT NULL, -- Sample: 1
  `properties_value` VARCHAR(100) NULL, -- Sample: Musical Instrument
  INDEX `fk_properties2item_properties1_idx` (`properties_properties_id` ASC) VISIBLE,
  PRIMARY KEY (`items_internal_id`, `items_variant_1`, `items_variant_2`, `properties_properties_id`),
  INDEX `fk_properties2item_items1_idx` (`items_internal_id` ASC, `items_variant_1` ASC, `items_variant_2` ASC) VISIBLE,
  CONSTRAINT `fk_properties2item_properties1`
    FOREIGN KEY (`properties_properties_id`)
    REFERENCES `mydb`.`properties` (`properties_id`)
  CONSTRAINT `fk_properties2item_items1`
    FOREIGN KEY (`items_internal_id` , `items_variant_1` , `items_variant_2`)
    REFERENCES `mydb`.`items` (`internal_id` , `variant_1` , `variant_2`)

CREATE TABLE hha  (
  `hha_id` INT NOT NULL AUTO_INCREMENT, -- Sample: 1
  `hha_name` VARCHAR(100) NOT NULL, -- Sample: hha_concept_1
PRIMARY KEY (`hha_id`))

CREATE TABLE hha2item (
  `items_internal_id` INT NOT NULL, -- Sample: 383
  `items_variant_1` INT NOT NULL, -- Sample: 0
  `items_variant_2` INT NOT NULL, -- Sample: 0
  `hha_hha_id` INT NOT NULL, -- Sample: 1
  `hha_value` VARCHAR(45) NULL, -- Sample: music
  INDEX `fk_item2hha_hha1_idx` (`hha_hha_id` ASC) VISIBLE,
```

```
  PRIMARY KEY (`items_internal_id`, `items_variant_1`, `items_variant_2`, `hha_hha_id`),
  INDEX `fk_item2hha_items1_idx` (`items_internal_id` ASC, `items_variant_1` ASC, `items_variant_2` ASC) VISIBLE,
  CONSTRAINT `fk_item2hha_hha1`
    FOREIGN KEY (`hha_hha_id`)
    REFERENCES `mydb`.`hha` (`hha_id`)
  CONSTRAINT `fk_item2hha_items1`
    FOREIGN KEY (`items_internal_id` , `items_variant_1` , `items_variant_2`)
    REFERENCES `mydb`.`items` (`internal_id` , `variant_1` , `variant_2`)


CREATE TABLE items (
  `internal_id` INT NOT NULL, -- Sample: 383
  `variant_1` INT NOT NULL, -- Sample: 0
  `variant_2` INT NOT NULL, -- Sample: 0
  `file_name` VARCHAR(255) NOT NULL, -- Sample: FtrAcorsticguitar_Remake_0_0
  `unique_entry_id` VARCHAR(255) NOT NULL, -- Sample: EpywQXABBcv2dipsP
  `item_name` VARCHAR(255) NOT NULL, -- Sample: acoustic guitar
  `variation` VARCHAR(100) NULL, -- Sample: Natural
  `body_title` VARCHAR(100) NULL, -- Sample: Body
  `version` VARCHAR(45) NOT NULL, -- Sample: 1.0.0
  `diy` TINYINT(3) NOT NULL, -- Sample: 1
  `body_customize` TINYINT(3) NOT NULL, -- Sample: 1
  `catalog` TINYINT(3) NULL, -- Sample: 0
  `outdoor` TINYINT(3) NOT NULL, -- Sample: 0
  `pattern_customize` TINYINT(3) NOT NULL, -- Sample: 0
  `hha_series` VARCHAR(45) NULL, -- Sample: antique
```

```
  `hha_set` VARCHAR(45) NULL, -- Sample: apple
  `source_source_id` INT NOT NULL, -- Sample: 4
  PRIMARY KEY (`internal_id`, `variant_1`, `variant_2`),
  UNIQUE INDEX `ıunique_entry_id_UNIQUE` (`unique_entry_id` ASC) VISIBLE,
  UNIQUE INDEX `file_name_UNIQUE` (`file_name` ASC) VISIBLE,
  INDEX `fk_items_source1_idx` (`source_source_id` ASC) VISIBLE,
  CONSTRAINT `fk_items_source1`
    FOREIGN KEY (`source_source_id`)
    REFERENCES `mydb`.`source` (`source_id`)
```

9-)

```
import pandas as pd
housewares_path = r".\housewares.csv"
housewares_df = pd.read_csv(housewares_path)
numerics_path = r".\numerics.csv"
numerics_df = pd.read_csv(numerics_path)
numerics2item_df = pd.DataFrame(columns=['internal_id', 'variant_1', 'variant_2', 'numerics_id', 'numerics_value'])
for index, row in housewares_df.iterrows():
    internal_id = row['internal_id']
    variant_1 = row['variant_1']
    variant_2 = row['variant_2']
    for numerics_index, numerics_row in numerics_df.iterrows():
        numerics_id = numerics_row['numerics_id']
        numerics_name = numerics_row['numerics_name']
        numerics_value = row[numerics_name]
        temp_df = pd.DataFrame({
            'internal_id': [internal_id],
```

```
          'variant_1': [variant_1],
           'variant_2': [variant_2],
           'numerics_id': [numerics_id],
           'numerics_value': [numerics_value]
        })
       numerics2item_df = pd.concat([numerics2item_df, temp_df],
ignore_index=True)
numerics2item_df.to_csv(r".\numerics2item.csv", index=False)


 color_path = r".\color.csv"
color_df = pd.read_csv(color_path)


color_dict = dict(zip(color_df['color_name'], color_df['color_id']))
result_df = pd.DataFrame(columns=['Internal ID', 'Variant_1', 'Variant_2',
'color'])
for index, row in housewares_df.iterrows():
   temp_df1 = pd.DataFrame({'Internal ID': [row['Internal ID']],
                            'Variant_1': [row['Variant_1']],
                            'Variant_2': [row['Variant_2']],
                           'color': [color_dict.get(row['Color 1'], None)]})


    temp_df2 = pd.DataFrame({'Internal ID': [row['Internal ID']],
                             'Variant_1': [row['Variant_1']],
                            'Variant_2': [row['Variant_2']],
                           'color': [color_dict.get(row['Color 2'], None)]})
   result_df = pd.concat([result_df, temp_df1, temp_df2], ignore_index=True)
 result_df.to_csv(r".\item2color.csv", index=False)
properties_path = r".\properties.csv"
```

```
properties_df = pd.read_csv(properties_path)
properties2item_df = pd.DataFrame(columns=['internal_id', 'variant_1', 'variant_2', 'properties_id', 'properties_value'])
for index, row in housewares_df.iterrows():
    internal_id = row['internal_id']
    variant_1 = row['variant_1']
    variant_2 = row['variant_2']
    for properties_index, properties_row in properties_df.iterrows():
        properties_id = properties_row['properties_id']
        properties_name = properties_row['properties_name']
        properties_value = row[properties_name]
        temp_df = pd.DataFrame({
            'properties_id': [properties_id],
            'internal_id': [internal_id],
            'variant_1': [variant_1],
            'variant_2': [variant_2],
            'properties_value': [properties_value]
        })

        properties2item_df = pd.concat([properties2item_df, temp_df], ignore_index=True)
properties2item_df.to_csv(r".\properties2item.csv", index=False)
```

These are the representative samples for our database. We used the same script for other tables only changing the variables.