**Database Management System Finals Documentation**

**BSIT 2-A**

**Group Members:**

**Santos, Francis Kyle D.**

**Mansilla, Anya Francine P.**

**Badic, Kim Angelo**

**Mateo, Emmaneul Walter D.**

**Interview Documentation**:

We interviewed a Vice President for Finance of the "Olanis Group of Companies, Inc." using the questionnaire approach. In the interview, the Vice President stressed the need to generate reports that are detailed yet actionable, which are basically important for their operations-at the main office and to the other branches. This data-driven methodology ensures better decision-making and easier management in the organization in general.

According to the interview, the following reports were suggested as potentially valuable options for students in developing a database and reporting system tailored for the company and possibly similar businesses:

* Branch Performance Report
  + Summarizing sales made on a daily, weekly, monthly, or yearly basis.
* Revenue Report
  + Tracking total earnings from different sources or operations over a defined timeframe, such as daily, weekly, monthly, or yearly.
* Inventory Report
  + Summarizing the current stock of tiles available at each branch, categorized by tile type and grouped by branches.
* Feedback Report
  + Summarizing and listing feedback received by the company within a specific month.

The following reports are proposed as useful options for the students to develop a database and reporting system customized for the company and perhaps similar businesses, considering the interview.

* Branch Performance Report
  + This report would summarize sales made on a daily, weekly, monthly, or yearly basis. It would give insight into how each branch is performing in terms of sales volume, revenue, and transaction count. The report can help identify the high-performing branches and areas for improvement.
* Revenue Report
  + Revenue reporting would maintain the record of the income generated through different means or operations during a stipulated period, say a day, week, month, or year. The company could then see its revenue performance, identify trends, and take appropriate business decisions in respect to business operations.
* Stock Report
  + This report would summarize how many tiles are in stock at any given time at each branch, by type and aggregate by branches. It would help the company manage the inventory level, realize which were the most popular products, and be certain of sufficient stock levels to satisfy demand.
* Feedback Report
  + The feedback report would summarize and list the customer feedback received by the company within a specific month. Customer ratings, comments, and suggestions improved customer services, products, and business performance in general. These reports would provide insights into all sorts of business aspects to make better decisions, operate efficiently, and satisfy customers.

**Expected Output:**

The reports that you create should clearly be displayed in a local browser, allowing the user to view the data dynamically. Efficient retrieval from the database will be essential, so queries should be optimized for faster performance, especially if there’s a lot of data. The PHP scripts will be responsible for querying the database and dynamically generating the report output, which will then be presented in the browser.

**Database Design:**

The SQL queries below were used to establish the desired database for the project as well as relevant tables in the database for holding data. Interrelation between tables was developed in such a manner that data was not likely to occur in inconsistency, and retrieval was done with utmost ease. The structure would lead to producing reports such as Branch Performance Report, Revenue Report, Inventory Report, and Feedback Report in accurate as well as in proper format. The following SQL queries are used to develop the database schema and ensure data integrity, which is being stored:

1. Database Creation:

create database if not exists Olanis;

use Olanis;

CREATE TABLE branches (

branch\_id INT AUTO\_INCREMENT PRIMARY KEY,

branch\_name VARCHAR(100) NOT NULL,

address TEXT NOT NULL,

contact\_number VARCHAR(15));

CREATE TABLE categories (

category\_id INT AUTO\_INCREMENT PRIMARY KEY,

category\_name VARCHAR(100) NOT NULL,

description TEXT);

CREATE TABLE tiles (

tile\_id INT AUTO\_INCREMENT PRIMARY KEY,

tile\_code VARCHAR(50) NOT NULL UNIQUE,

tile\_name VARCHAR(100) NOT NULL,

category\_id INT,

color VARCHAR(50) NOT NULL,

material VARCHAR(50),

tile\_size VARCHAR(50),

price DECIMAL(10, 2) NOT NULL,

cost\_price DECIMAL(10, 2) NOT NULL,

stock\_quantity INT DEFAULT 0,

branch\_id INT,

FOREIGN KEY (category\_id) REFERENCES categories(category\_id),

FOREIGN KEY (branch\_id) REFERENCES branches(branch\_id));

CREATE TABLE customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_name VARCHAR(100) NOT NULL,

contact\_number VARCHAR(15),

email VARCHAR(100),

address TEXT,

business\_permit VARCHAR(50),

referred\_by VARCHAR(100),

credit\_limit DECIMAL(10, 2) DEFAULT 0);

CREATE TABLE sales (

sale\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

tile\_id INT,

sale\_date DATE NOT NULL,

quantity INT NOT NULL,

total\_price DECIMAL(10, 2) NOT NULL,

branch\_id INT,

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),

FOREIGN KEY (tile\_id) REFERENCES tiles(tile\_id),

FOREIGN KEY (branch\_id) REFERENCES branches(branch\_id));

CREATE TABLE inventory (

inventory\_id INT AUTO\_INCREMENT PRIMARY KEY,

branch\_id INT,

tile\_id INT,

current\_stock INT NOT NULL,

last\_updated DATE NOT NULL,

FOREIGN KEY (branch\_id) REFERENCES branches(branch\_id),

FOREIGN KEY (tile\_id) REFERENCES tiles(tile\_id));

CREATE TABLE IF NOT EXISTS feedbackreports (

feedback\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

branch\_id INT,

feedback\_date DATE NOT NULL,

feedback\_text TEXT NOT NULL,

rating INT CHECK (rating BETWEEN 1 AND 5),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id),

FOREIGN KEY (branch\_id) REFERENCES branches(branch\_id));

1. Populate Tables with Data:

INSERT INTO branches (branch\_name, address, contact\_number) VALUES

('Main Branch', '6535+97P, National Highway, Barangay Real, Calamba, Laguna', '(049) 834 6978'),

('North Branch 1', '123 Main St, Metro City', '(049) 905 4132'),

('North Branch 2', '456 North St, Metro City', '(049) 762 0541');

INSERT INTO categories (category\_name, description) VALUES

('3D Tiles', 'Modern tiles with three-dimensional effects for creative designs'),

('Natural Stone Tiles', 'Eco-friendly tiles made from natural stone'),

('Vintage Tiles', 'Retro-style tiles for timeless aesthetics'),

('Eco-Friendly Tiles', 'Sustainable tiles made with recycled materials'),

('Anti-Slip Tiles', 'Safe tiles designed to prevent slipping');

INSERT INTO tiles (tile\_code, tile\_name, category\_id, color, material, tile\_size, price, cost\_price, stock\_quantity, branch\_id) VALUES

('3DT-015', 'Geometric Gray 3D Tile', 1, 'Gray', 'Porcelain', '45x45 cm', 14.00, 9.50, 120, 1),

('NAT-016', 'Rustic Natural Stone', 2, 'Earth', 'Stone', '50x50 cm', 18.00, 12.50, 100, 2),

('VIN-017', 'Vintage Floral Pattern', 3, 'Blue', 'Ceramic', '30x30 cm', 10.00, 6.00, 150, 3),

('ECO-018', 'Recycled Green Tile', 4, 'Green', 'Ceramic', '30x30 cm', 8.00, 5.50, 200, 1),

('ANT-019', 'Textured Anti-Slip', 5, 'Black', 'Porcelain', '60x60 cm', 12.00, 7.00, 90, 2),

('3DT-020', 'Wavy White 3D Tile', 1, 'White', 'Porcelain', '45x90 cm', 16.00, 10.00, 80, 3),

('NAT-021', 'Smooth River Stone', 2, 'Gray', 'Stone', '60x40 cm', 20.00, 13.50, 60, 1),

('VIN-022', 'Checkerboard Vintage Tile', 3, 'Black and White', 'Ceramic', '20x20 cm', 9.00, 5.00, 180, 2),

('ECO-023', 'Eco-Friendly Brown', 4, 'Brown', 'Porcelain', '30x30 cm', 7.50, 4.50, 250, 3),

('ANT-024', 'Rough Texture Safety Tile', 5, 'Red', 'Ceramic', '60x60 cm', 15.00, 9.00, 70, 1);

INSERT INTO customers (customer\_name, contact\_number, email, address, business\_permit, referred\_by, credit\_limit) VALUES

('Chris Black', '111-222-3333', 'chris.black@example.com', '777 Fern St, Metro City', 'BP111222', NULL, 4500.00),

('Patricia Gray', '444-555-6666', 'patricia.gray@example.com', '888 Ash St, Metro City', 'BP444555', 'Sophia Blue', 5500.00),

('Liam Gold', '777-888-9999', 'liam.gold@example.com', '999 Cedar St, Metro City', 'BP777888', 'John Doe', 6000.00),

('Amelia Silver', '888-999-0000', 'amelia.silver@example.com', '1010 Iron St, Metro City', 'BP888999', 'Jane Smith', 5000.00),

('Lucas Pearl', '999-000-1111', 'lucas.pearl@example.com', '1111 Bronze St, Metro City', 'BP999000', 'Alice Green', 3000.00);

INSERT INTO sales (customer\_id, tile\_id, sale\_date, quantity, total\_price, branch\_id) VALUES

(2, 1, '2024-12-24', 25, 350.00, 1),

(4, 2, '2024-12-25', 10, 180.00, 2),

(5, 3, '2024-12-26', 20, 200.00, 3),

(3, 4, '2024-12-27', 15, 120.00, 1),

(1, 5, '2024-12-28', 12, 144.00, 2);

INSERT INTO inventory (branch\_id, tile\_id, current\_stock, last\_updated)

VALUES

(1, 1, 95, '2024-12-24'),

(2, 2, 90, '2024-12-25'),

(3, 3, 130, '2024-12-26'),

(1, 4, 185, '2024-12-27'),

(2, 5, 78, '2024-12-28');

INSERT INTO feedbackreports (customer\_id, branch\_id, feedback\_date, feedback\_text, rating) VALUES

(3, 1, '2024-12-24', 'Loved the 3D tiles, very modern!', 5),

(4, 2, '2024-12-25', 'Natural stone tiles are beautiful.', 5),

(2, 3, '2024-12-26', 'Vintage tiles were perfect for my home.', 4),

(1, 1, '2024-12-27', 'Eco-friendly tiles are a great concept!', 5),

(5, 2, '2024-12-28', 'Anti-slip tiles work great in bathrooms.', 4);

1. SQL Functions For Each Report
   1. Branch performance report

SELECT branches.branch\_name, SUM(sales.total\_price) AS total\_sales, COUNT(sales.sale\_id) AS total\_transactions, AVG(sales.total\_price) AS avg\_sale\_value, SUM(sales.quantity) AS products\_sold FROM sales INNER JOIN branches ON sales.branch\_id = branches.branch\_id GROUP BY branches.branch\_name ORDER BY total\_sales DESC;

* 1. Revenue Report
  2. Inventory Report

SELECT b.branch\_name AS 'Branch Name', t.tile\_name AS 'Tile Name', c.category\_name AS 'Category', i.current\_stock AS 'Stock Quantity', i.last\_updated AS 'Last Updated' FROM inventory i JOIN branches b ON i.branch\_id = b.branch\_id JOIN tiles t ON i.tile\_id = t.tile\_id JOIN categories c ON t.category\_id = c.category\_id ORDER BY b.branch\_name, t.tile\_name;

* 1. Feedback Report