DBMS Project Report

Amardeep Singh Bhullar

CSCI 126

Student Number: 109963782

Dr. David Ruby

13-May-2022

Table of Contents

Introduction

I am Amardeep Singh Bhullar. This report encapsulates all the work I during the DBMS project assigned during the course CSCI 126. The project pushed me to think laterally and understand the skills to create my own Relational Database. Furthermore, I learned how to create Database(s), Schema(s), Tables, drawing data from real data sources like- www.kaggle.com. I am extremely obliged to have taken this course, as it helped me to understand the concepts of Database Management better. A big thank you to Dr. David Ruby for helping me every step of the way.

Key Definitions

- **Database-** A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS).

- **Relational databases-** Items in a relational database are organized as a set of tables with columns and rows. Relational database technology provides the most efficient and flexible way to access structured information.

- **Database schema-** is a visual representation of database. The structure is described by formal language and is supported by database management system (DBMS). It represents logical view of the database.

- **Query-** is a request for data or information from a database table or combination of tables. This data may be generated as results returned by Structured Query Language (SQL) or as pictorials, graphs, or complex results, e.g., trend analyses from data-mining tools.

- **Structured Query Language (SQL)**- is a programming language that is typically used in relational database or data stream management systems.

- **Entity Relationship (ER) Diagram-** is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system.

Data Used

The data set used in this project contains several products and real customer search terms from Home Depot's website, managing: -

- **Employee details:** personnel and professional details.
- **Product details:** Product inventory
- **Customer details:** details- their address, payment, billing, and shipping information.

Tables

- **Employee: includes employee name, their personal information, and their service/engagement for/with the company.**

```
CREATE TABLE Employee
(
Department varchar(40),
Age int,
Employee_Number int NOT NULL,
Employee_Type varchar(20),
Gender varchar(6),
Monthly_Income int,
Job_Role varchar(40),
YearsAtCompany int,
PRIMARY KEY (Employee_Number)
);
CREATE INDEX idx_employee_emplnum ON Employee (Employee_Number);
```

- **Customer: consists customer information- name and email address.**

```
CREATE TABLE Customer
(
Customer_ID varchar(12) NOT NULL,
First_Name varchar(20),
Last_Name varchar(20),
Email_Address varchar(36),
PRIMARY KEY (Customer_ID)
);
```

- **Reviews: shows reviews by the customers.**

```
CREATE TABLE Reviews
(
Rating int,
Review_ID varchar(10) NOT NULL,
Product_ID varchar(7) NOT NULL,
PRIMARY KEY (Review_ID)
);
```

- **Payment: payment details of the customers.**

```
CREATE TABLE Payment
(
Payment_ID varchar(10) NOT NULL,
Payment_Method varchar(8),
CCN int,
Card_NAME varchar(35),
Customer_ID varchar(12) NOT NULL,
PRIMARY KEY (Payment_ID),
CONSTRAINT `Creates` FOREIGN KEY (`Customer_ID`) REFERENCES
`Customer` (`Customer_ID`) ON DELETE RESTRICT ON UPDATE RESTRICT
);
CREATE INDEX idx_payment_customer ON Payment (Customer_ID);
```

- **Grouping: Grouping products**

```
CREATE TABLE Product_Grouping
(
Category varchar(10),
Grouping_ID varchar(15) NOT NULL,
PRIMARY KEY (Grouping_ID)
);
```

- **Orders: Order details like payment and status details.**

```
CREATE TABLE Orders
(
Order_ID varchar(50) NOT NULL,
Order_Date varchar(40),
Payment_ID varchar(10) NOT NULL,
Status varchar(20),
PRIMARY KEY (Order_ID),
CONSTRAINT `Orders` FOREIGN KEY (Payment_ID) REFERENCES
`Payment` (Payment_ID) ON DELETE RESTRICT ON UPDATE RESTRICT
);
CREATE INDEX idx_orders_payment ON Orders (Payment_ID);
```

- **Product: Listing all the products (Inventory).**

```
CREATE TABLE Product
```

```
(
Product_ID varchar(7) NOT NULL,
Product_Name varchar(80),
Price float,
User_ID varchar(40),
Review_ID varchar (10) NOT NULL,
Order_ID varchar(50) NOT NULL,
Grouping_ID int NOT NULL,
PRIMARY KEY (Product_ID)
FOREIGN KEY (Grouping_ID) REFERENCES
`Product_Grouping` (Grouping_ID) ON DELETE RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (Review_ID) REFERENCES
`Reviews` (Review_ID) ON DELETE RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (Order_ID) REFERENCES
`Orders` (Order_ID) ON DELETE RESTRICT ON UPDATE RESTRICT
);
CREATE INDEX idx_product_grouping ON Product (Grouping_ID);
CREATE INDEX idx_product_reviewID ON Product (Review_ID);
```

- **Address: Customer address details.**

```
CREATE TABLE Address
(
Address_Hash varchar(50) NOT NULL,
Customer_ID varchar(12) NOT NULL,
PRIMARY KEY (Address_Hash),
CONSTRAINT `has` FOREIGN KEY (Customer_ID) REFERENCES
`Customer` (Customer_ID) ON DELETE RESTRICT ON UPDATE RESTRICT
);
CREATE INDEX idx_address_customer ON Address (Customer_ID);
```

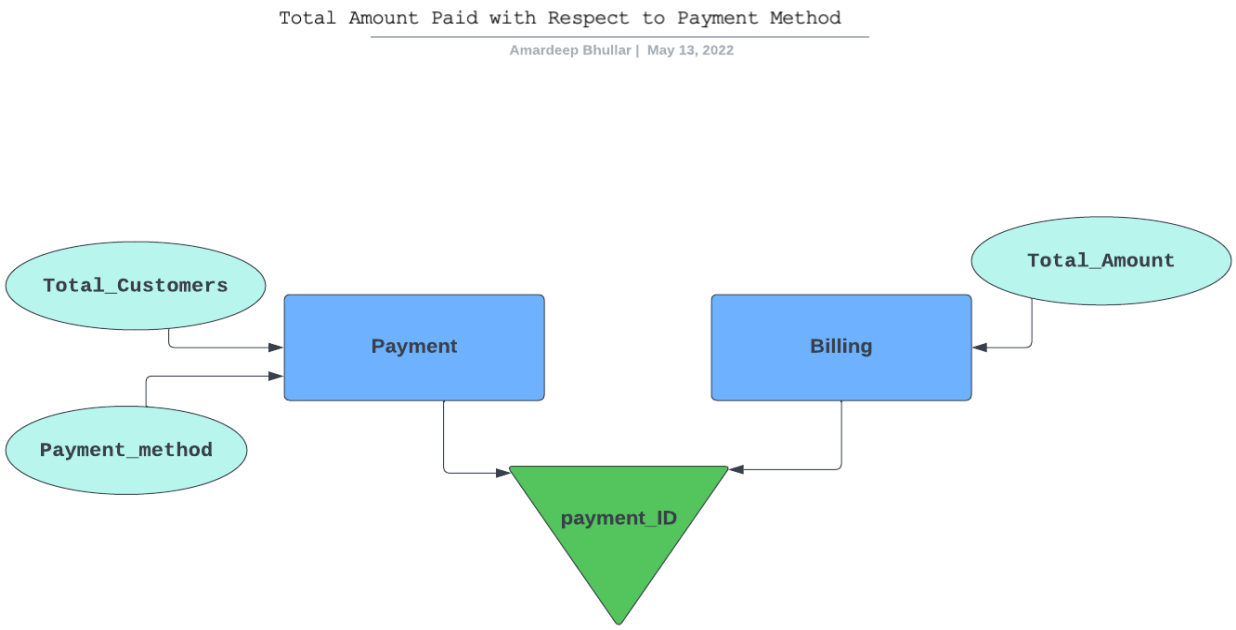
- **Billing: Billing details of the customers.**

```
CREATE TABLE Billing
(
Billing_ID varchar(50) NOT NULL,
Product_ID varchar(7) NOT NULL,
Amount float,
Payment_ID varchar(10) NOT NULL,
Order_ID varchar(50) NOT NULL,
PRIMARY KEY (Billing_ID),
CONSTRAINT `Makes` FOREIGN KEY (Payment_ID) REFERENCES
`Payment` (Payment_ID) ON DELETE RESTRICT ON UPDATE RESTRICT,
CONSTRAINT `Is Billed` FOREIGN KEY (Order_ID) REFERENCES
`Orders` (Order_ID) ON DELETE RESTRICT ON UPDATE RESTRICT
);
CREATE INDEX idx_billing_payment ON Billing (Payment_ID);
CREATE INDEX idx_billing_order ON Billing (Order_ID);
```

ER Diagrams for Queries

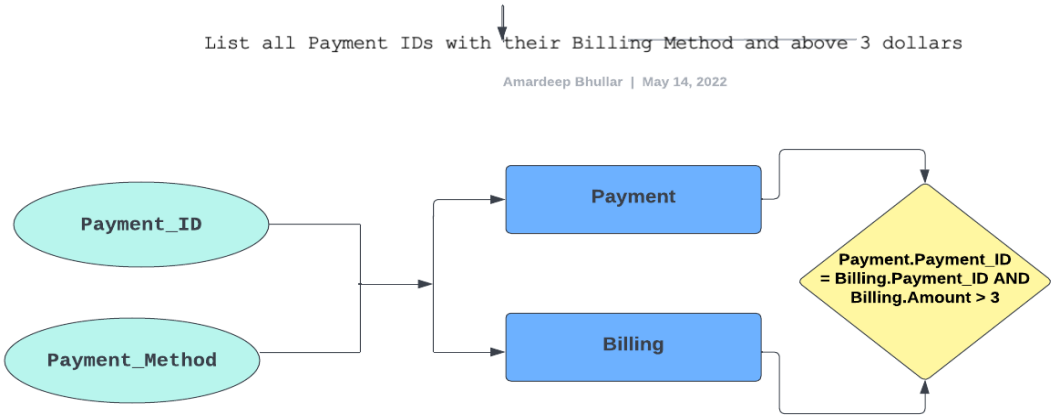
1. Total Amount Paid with Respect to Payment Method

```
SELECT count(Payment_Method) AS Total_Customers, Payment.Payment_Method,
SUM(Billing.Amount)
AS Total_Amount FROM ProjectDB.Payment INNER JOIN ProjectDB.Billing ON
Payment.Payment_ID =
Billing.Payment_ID GROUP BY Payment_Method;
```



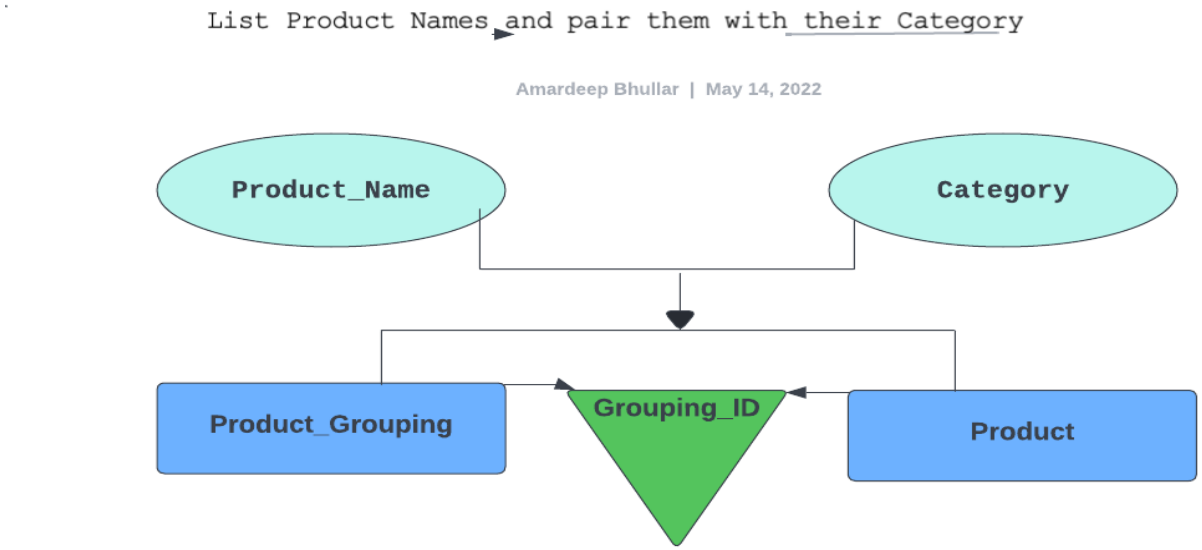
2. List all Payment IDs with their Billing Method and above 3 dollars.

```
SELECT Payment.Payment_ID, Payment.Payment_Method FROM Billing, Payment
WHERE Payment.Payment_ID = Billing.Payment_ID AND Billing.Amount > 3;
```



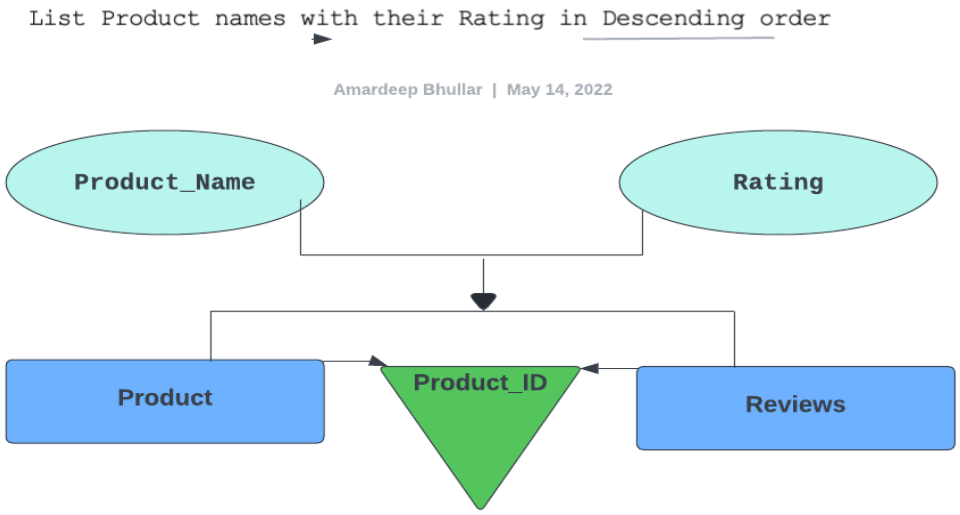
3. List Product Names and pair them with their Category.

```
SELECT Category, Product_Name FROM Product_Grouping INNER JOIN Product ON
Product_Grouping.Grouping_ID = Product.Grouping_ID ORDER BY Category;
```



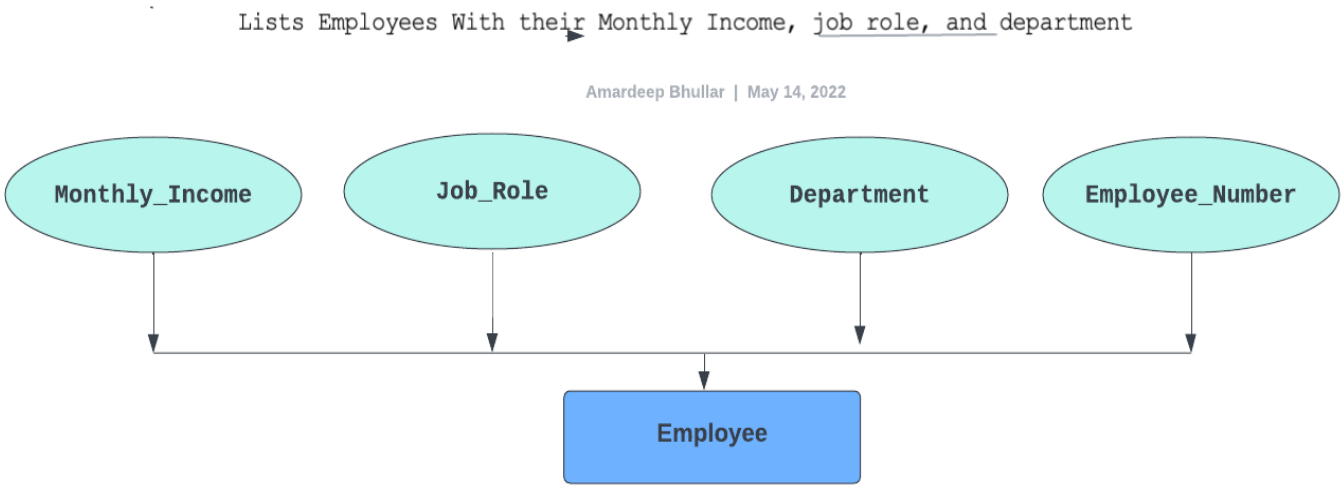
4. List Product names with their Rating in Descending order

```
SELECT Product_Name, Rating FROM Product INNER JOIN Reviews ON
Reviews.Product_ID = Product.Product_ID ORDER BY Rating DESC;
```



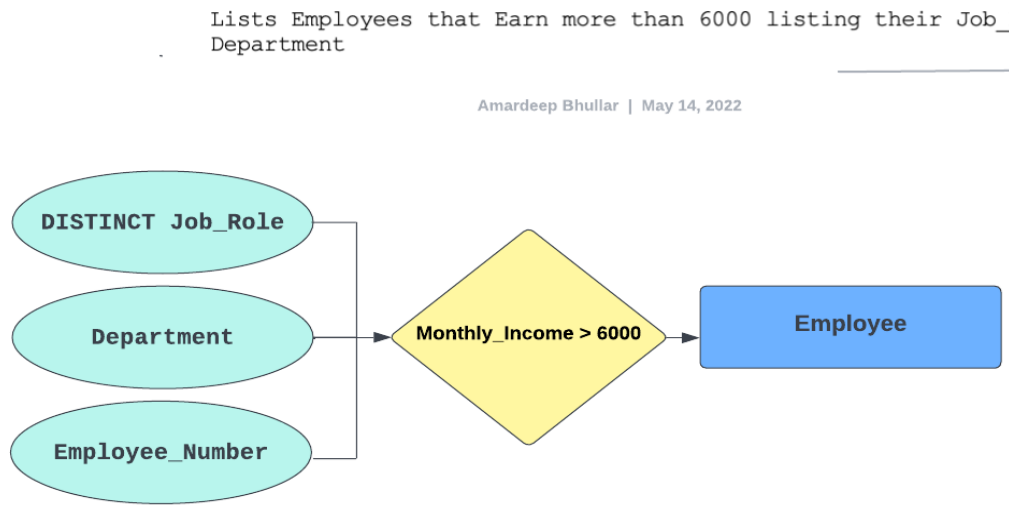
5. Lists Employees With their Monthly Income, job role, and department.

```
SELECT Monthly_Income, Job_Role, Department, Employee_Number FROM Employee
ORDER BY Monthly_Income DESC;
```



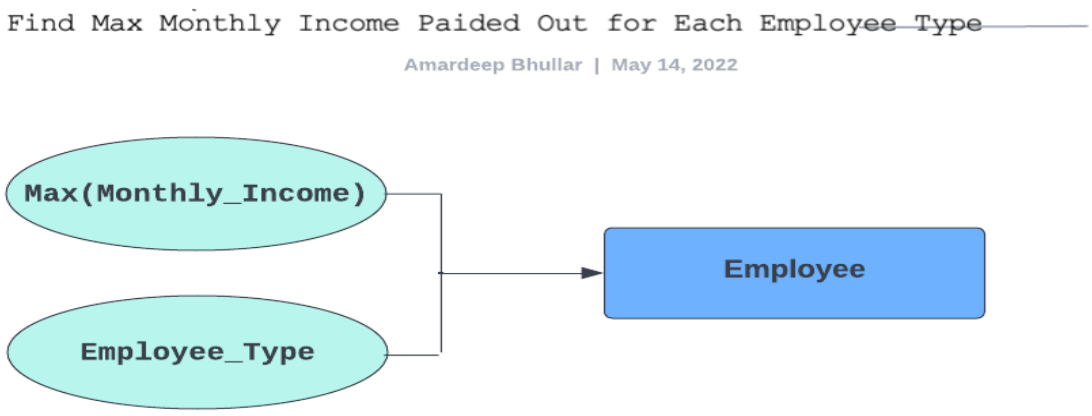
6. Lists Employees that Earn more than 6000 listing their Job_Role and Department.

```
SELECT DISTINCT Employee.Job_Role, Employee.Employee_Number,
Employee.Department FROM Employee WHERE Employee.Monthly_Income > 6000
LIMIT 6;
```



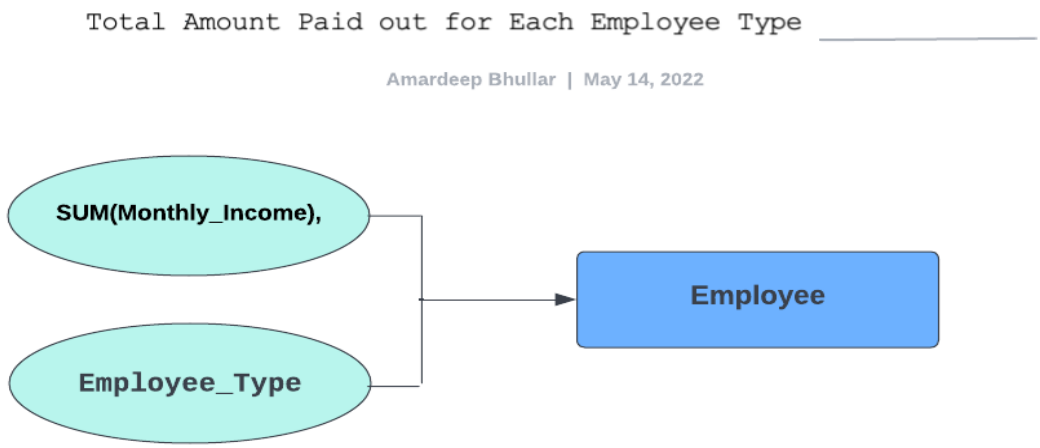
7. Find Max Monthly Income Paided Out for Each Employee Type

```
SELECT MAX (Monthly_Income), Employee_Type FROM Employee GROUP BY
Employee_Type;
```



8. Total Amount Paid out for Each Employee Type

```
SELECT SUM(Monthly_Income), Employee_Type FROM Employee GROUP BY
Employee_Type;
```



Bibliography

https://www.oracle.com/index.html

https://www.techopedia.com/

https://www.kaggle.com/

https://www.lucidchart.com/

https://www.homedepot.com/