Database Management and Design

Lesson #1

Course Goals

To be able to

* Design a well-designed relational database
* Verify the design of a database by applying the rules of normalization and data modeling checklist
* Create an E-R diagram to present a database design using a diagramming tool such a MySQL Workbench
* Implement a relational database using the SQL data definition language (DDL) or SSMS GUI
* Retrieve data and information from a relational database using SQL data manipulation language (DML)

Optional Course Goals

To be able to

* Describe transaction processing and explain the complexity it adds to data processing
* To write triggers and/or procedures with T-SQL to implement business requirements and/or to enforce database consistency
* Connect a SQL database to a front end written in a host language such as Java

Lesson 1 - Behavioral Objectives

Student will be able to

* Identify the basic characteristics of a relational table
* Identify which column(s) can be most relied upon to store a unique value in each record and therefore uniquely identify an instance of the entity stored in that table (***primary key***)
* Identify which column(s) is/are used to link related records within a relational database (***foreign key***)
* Given certain criteria, identify which records match the given criteria
* Compute aggregate values of data stored in a table.
* Identify columns that contain data that is derived from computations based on data in other columns of the same or different tables.(***computed columns***)
* Identify data that should be unique within a data set. (**unique constraint**)
* Enumerate the basic components of a relational database and rules of a relational table
* Identify data redundancy and inconsistencies within a table
* Examine an ERD and identify the basic business rules and entity relationships it presents.
* Define database related terms such as primary key, foreign key, domain constraint, candidate key, referential integrity, entity integrity, null value
* Identify and describe the update, delete and insertion anomalies that exist in a poorly designed table
* Understand how to navigate and retrieve information from a set of related relational tables

Premiere company database **PremiereCo**

**Characteristics of a relational table**

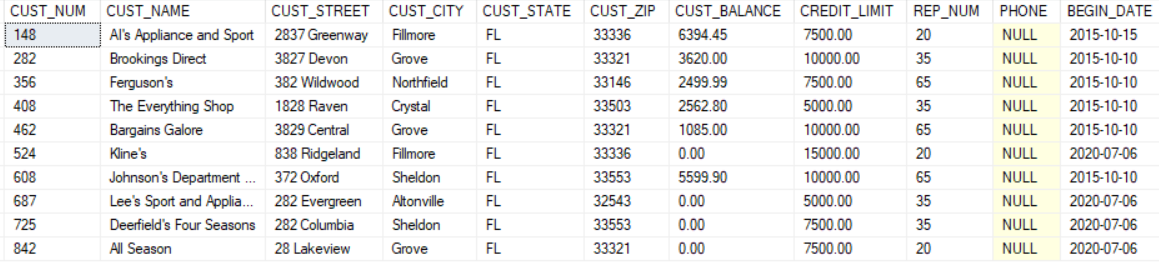
1. Can be viewed as a matrix - collection of rows and columns
2. Each table row represents one entity within a set of like entities
3. Each table column represents an attribute and each column has a unique name
4. Each cell (intersection of a row and a column) contains one single data value (**no multi-valued attributes**)
5. All values within a column must be of the same data format (data type)
6. Each column has a specific range of values (**attribute domain**)
7. The order of the rows and columns makes no difference
8. Each table must have an attribute or combination of attributes that uniquely identifies each row (**primary key**)

Well-designed relational table

1. The primary key must **determine** each attribute in a record. This means that given one value of the primary key, looking through the table you will find only one value for each of the columns within the table.
2. The table should not contain any **repeating groups** of columns or **multi-valued attributes**.
3. Besides the primary key or an alternate column that could be a primary key (**candidate key**) no other column should determine other attributes within the table.

**The following table is well designed**

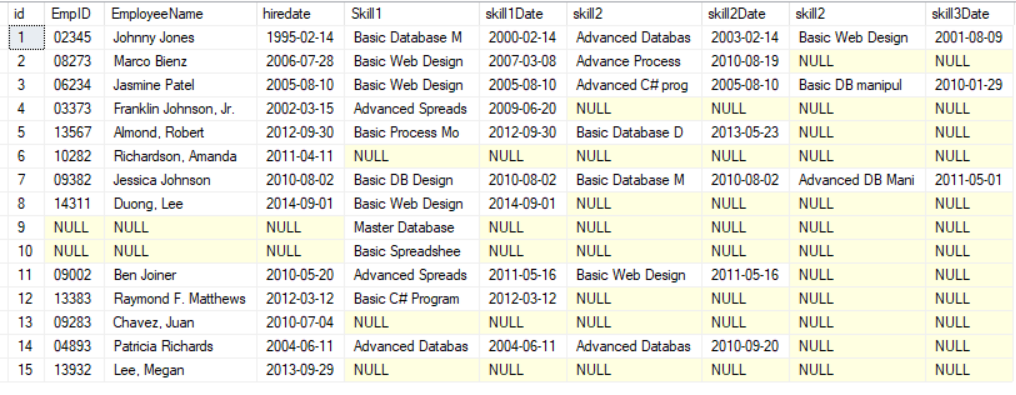
**Answer Questions 1-3 on Canvas:**



**The following table is NOT well designed**

**Reasons:**

1. Table includes information about two different entities, Employees and Skills
2. Table includes a repeating group
3. The repeating group has introduced many unnecessary **null values** into the table
   * Null values are greatly discouraged: waste memory, skew aggregate results, can be misinterpreted
4. This design will make searching and sorting, ensuring data consistency and the elimination of inconsistencies cumbersome and time consuming
   * We will explore this in more detail throughout the semester

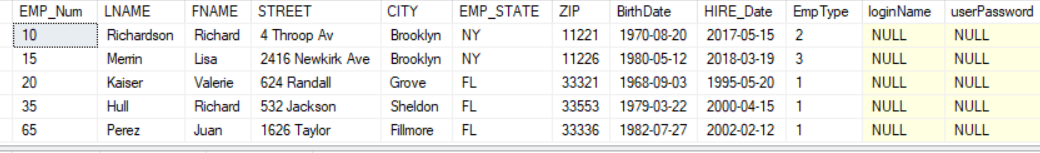
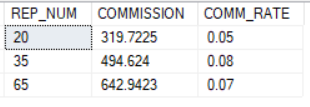


**Explore the relationship between two closely well designed, related tables:**

**a Salesrep “is an” Employee**

1. We divide the data into two tables since commission and commission rate only applies to Sales People, and not to other Employees
2. The tables must share something in common that is unique so that the data can be linked together. What is the link?
3. **Referential integrity:** will prevent a user from inserting a record into the salesrep table for an employee whose emp\_num doesn’t already appear in the Employee table. Why is this important?

**Employee Table**



**Salesrep table**

Questions: Study the Employee and SalesRep tables

1. What is the name of the salesperson whose ID is 20? How did you determine this information? (**concept: foreign key**)

Answer: The name of the salesperson is Valerie Kaiser. The REP\_NUM column works as a foreign key in the SalesRep table, and the EMP\_NUM column, the primary key in the employee table, is the same, which is 20.

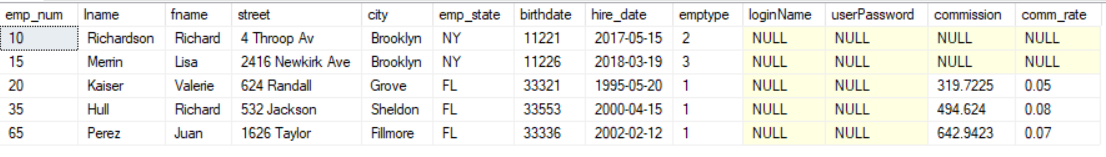
1. Is every Employee a salesperson? How did you determine this? (**concept: difference operation**)

Answer: No, Every Employee is not a salesperson. As we can see, the types of employees are different and employees number 10 and 15 don’t get a commission, which the others do. That determines that every employee is not a salesperson.

1. What information is relevant to a salesperson but not relevant to other employees?

Answer: Commission-based metrics, such as COMMISSION and COMM\_RATE, are critical pieces of information for salespeople. These measures directly affect their revenues and are critical to their work. Other employees, such as administrative or managerial personnel, would normally not require this information unless they also have commission-based income or performance targets.

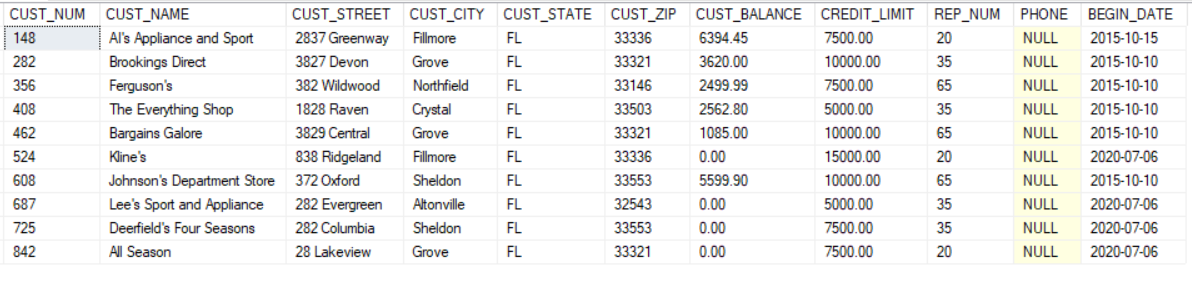
1. (see table below) What would happen if we decided to store the Commission and Comm\_Rate data in the Employee table, instead of setting up a separate Salesrep table? Why is this not such a good idea? (**concept: nulls, pk should determine each field**) Why are the loginName and userPassword columns included in the Employee table even though they currently contain nulls?



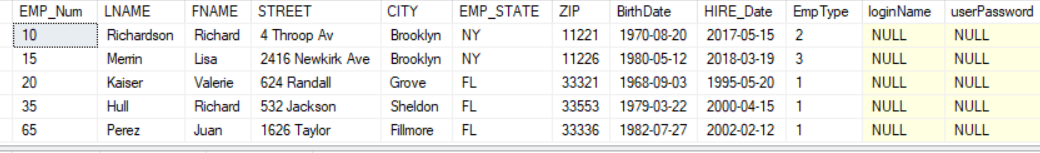
Answer: Storing commission and commission rate data in the Employee table is not recommended because it results in null values for non-sales employees, wasting space and potentially causing confusion. The Salesrep table is a more standard solution because it just stores commission data for salespeople, which eliminates redundancy and inefficiencies.

The loginName and userPassword columns are most likely included to future-proof the database, as all employees may require access to a system, even if some do not now use it. They are most likely designed to provide flexibility and consistency in how employee credentials are managed across roles.

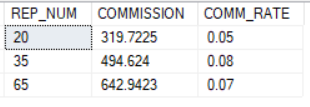
**Customer table**



**Employee table**



**Salesrep table**



Questions: Study the Customer, SalesRep and Employee tables

1. What is the name of the sales rep who services Al’s Appliance and Sport? How did you determine this? (**concept: foreign key**)

Answer: The name of the sales rep who services Al’s Appliance and Sport is Valerie Kaiser. The REP\_NUM column which works as a foreign key in the customer table and works as a primary key in the sales rep table are same which is 20.

1. Which field besides for cust\_num could possibly uniquely identify each Customer? (**concept: candidate key**) Can a primary key or candidate key allow nulls? Why?

Answer: Besides CUST\_NUM, the PHONE field could potentially serve as a candidate key, assuming all customers have unique phone numbers. Other factors such as a combination of CUST\_NAME, CUST\_STREET, CUST\_CITY, and CUST\_ZIP may be able to uniquely identify customers, but CUST\_NUM is most likely the sole reliable field for uniqueness.

A primary key and a candidate key cannot accept null values because they are intended to uniquely identify entries, and null values would undermine this uniqueness and cause data integrity issues.

1. Can the cust\_name be used to uniquely identify each Customer? What would have to be true about data in the cust\_name column to use this column to uniquely identify each Customer?(**concept: primary key**)

Answer: No, CUST\_NAME cannot be used to identify each client individually because names are rarely unique in a database.

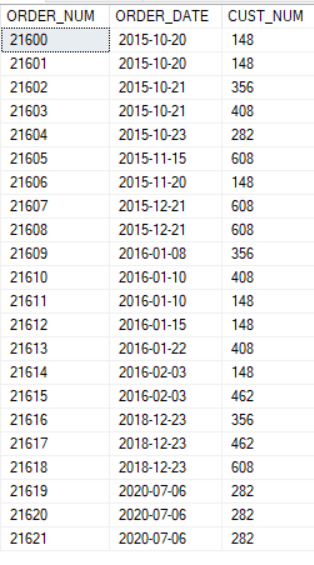
To utilize CUST\_NAME as a primary key, each name must be unique, non-null, and immutable, which is both restrictive and unworkable in real-world applications. Primary keys should be stable and unique, which is why CUST\_NUM (a system-generated, unique identifier) is often the ideal choice for a primary key because it avoids the problems associated with name duplication and modifications.

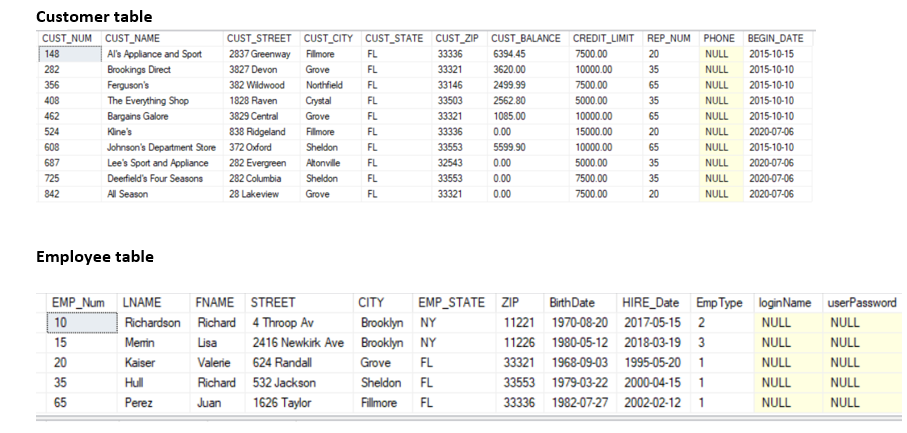
1. What is the relationship between the CREDIT\_LIMIT and CUST\_BALANCE fields? (**concept: business rules, constraints**)

Answer: The link between CREDIT\_LIMIT and CUST\_BALANCE is critical in regulating a customer's credit utilization. The CREDIT\_LIMIT limits how much the consumer can owe, whereas the CUST\_BALANCE shows how much they owe.

Business rules prevent the balance from exceeding the credit limit, and constraints (such as CHECK constraints or triggers) can enforce these rules within a database.

If the CUST\_BALANCE approaches or surpasses the CREDIT\_LIMIT, firms often take measures to mitigate the risk, such as sending payment reminders, prohibiting additional transactions, or changing credit conditions.

**Orders Table**



Questions: Study the Orders, Customer and Employee tables

1. What is the name of the customer who placed the order: 21606? (**foreign key**) How about order 21618?

Answer: The name of the customer who placed the order: 21606 is AI's Appliance and Sport. The CUST\_NUM 148, which works as a foreign key in the Orders table is the primary key in the Customer table.

Similarly, the name of the customer who placed the order: 21618 is Johnson's Department Store. The CUST\_NUM 608, which works as a foreign key in the Orders table is the primary key in the Customer table.

1. Name the salesperson who took care of placing the order: 21606. How did you figure this out?

Answer: The salesperson who took care of placing order 21606 is Valerie Kaiser. First, the relationship between the customer table and the order table is checked, where the CUST\_NUM 148 works as a foreign key in the order table and the primary key in the customer table. Then, the relationship between the customer table and the employee table is checked, where the EMP\_NUM 20 works as a foreign key in the customer table and the primary key in the employee table.

1. How many orders did Customer Al’s Appliances and Sport place? (**concept: aggregate function**)

Answer: SQL Query:

SELECT COUNT(ORDER\_NUM) AS num\_orders

FROM orders

WHERE CUST\_NUM = 148;

AI's Appliance and Sport placed six orders.

1. How many orders did each Customer place? (**concept: aggregate function**)

Answer: SQL Query:

SELECT CUST\_NUM, COUNT(ORDER\_NUM) AS num\_orders

FROM orders

GROUP BY CUST\_NUM

ORDER BY num\_orders DESC;

To determine how many orders each customer placed, we can use the COUNT() aggregate function and group the data by CUST\_NUM.Based on the above query we got the following result.

A computer screen shot of a computer

AI-generated content may be incorrect.

**Poorly Designed Tables**

Let us explore what would happen if we decided to include the Orders information in the Customer table?

* 1. How easily could we determine which customer placed order 21623? (**concept: repeating group**) How about order 21616 or order 21606?

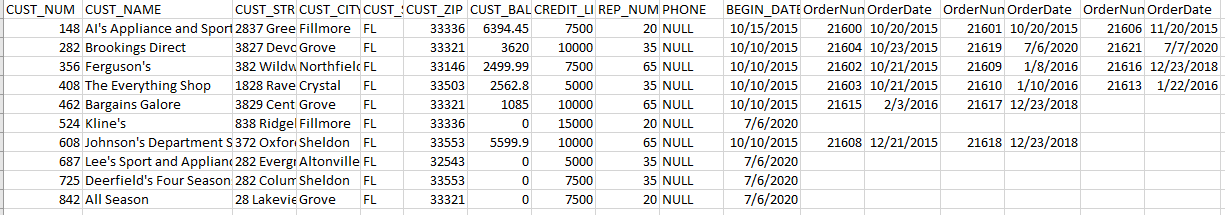
Answer: Because of the redundant data and the need to scan through rows, storing orders directly in the customer table makes it more difficult to determine which customer placed a certain order (such as 21623, 21616, or 21606). This causes inefficiencies and potential data integrity issues.

* 1. How easily can we determine how many orders each Customer placed?

Answer: Even if orders are kept in the customer database (repeating groups), you may still count the rows for each customer to determine how many orders they placed. However, this can become inefficient as the table expands.

* 1. What will happen to Orders 21600, 21601, and 21606 if we delete Customer 148 from our database? (**deletion anomaly**)

Answer: A deletion anomaly happens when deleting one record (such as a customer) accidentally results in the loss of associated data (such as orders) that should not be destroyed. Repeating groups (in which orders are stored directly in the customer table) worsens the problem because deleting a customer deletes all of their connected orders due to a lack of separation between the entities.



Let us explore what would happen if we decided to include the salesrep information in the Customer table?

* 1. How many times have we stored Valerie Kaiser’s birthdate, hiredate and login id? (**concept: redundancy**)

Answer: Valerie Kaiser's birthdate, hire date, and login ID are all included in the Employee table.

They are not stored in the Customer or SalesRep tables.

Answer:

Birthdate: one time (in the Employee table).

Hire date: one time (in the Employee table)

Login ID: 0 times (currently nil, so not stored)

Valerie Kaiser has these values stored twice. Her login ID is currently null, therefore there is no redundancy.

* 1. How would we enter data about the Salesrep, if we hire a new Salesrep but haven’t yet assigned any Customer to that Salesrep ? (**concept:** **insertion anomaly**)

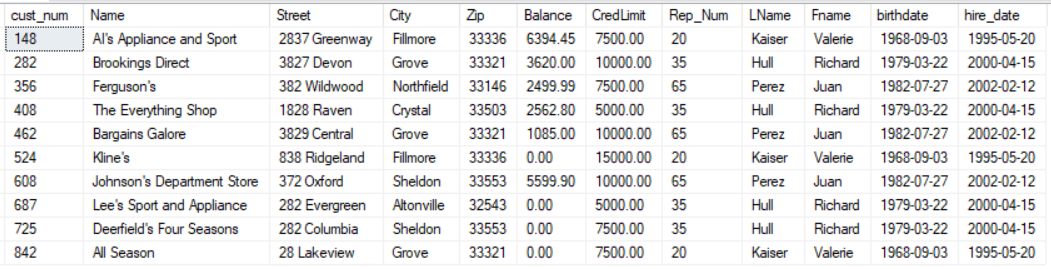
Answer: If SalesRep data is kept directly in the Customer database, you will encounter an insertion anomaly because you cannot add SalesRep data unless a customer is associated with it. You have to create mock customer records to link to the SalesRep (not recommended). Alternatively, wait until a client is allocated to the new SalesRep before inputting their information.

* 1. If Valerie gets married and decides to change her last name, how many records must be update? (**concept: update anomaly**)

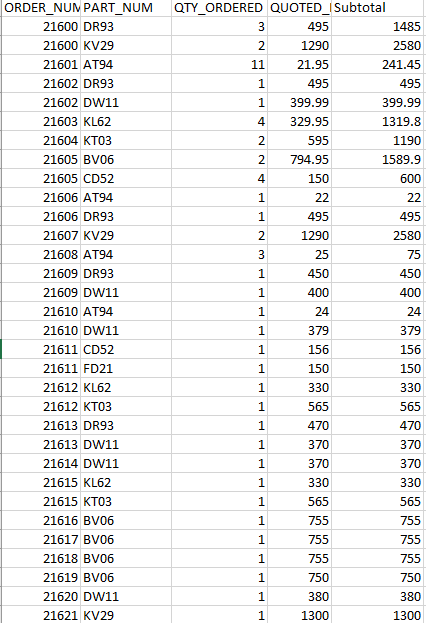
Answer: If Valerie changes her last name, two records will need to be modified (one in the Employee table and one in the Customer table). This is an example of an update anomaly induced by data redundancy between many tables.

* 1. If we don’t update Valerie’s last name in each of the records, will we be able to determine what Valerie’s last name is? (**concept: data inconsistency**)

Answer: If we do not update Valerie's last name in all records where it appears (i.e., in the Employee and Customer tables), we will have data inconsistency. As a result, there is uncertainty about her actual last name, which can lead to confusion, errors, and more complex inquiries to address the inconsistency.



**Order\_line Table**



Questions: Study the Orders , Order\_Line, Customer, SalesRep tables

1. Which customers ordered part\_num DR93? (**concept: foreign key**)

Answer: AI's Appliance and Sport ordered the part DR93. The ORDER\_NUM 21600, which acts as the primary key in the Order table and acts as a foreign key in the order\_line table shows the relationship between the two tables. Then, CUST\_NUM 148 acts as the primary key in the Customer table and the foreign key in the order table that shows the relationship between the two tables.

1. Which column in Order\_Line is derived from information in another column or columns of that table? (**concept: derived field**) How can we ensure that this data remains consistent with the data in the other column or columns?

Answer: The Subtotal field in the Order\_Line table is derived since it is determined using the QTY\_ORDERED and QUOTED\_PRICE variables. To ensure that the Subtotal matches these other fields, you can:

* Set up a trigger to update the Subtotal if QTY\_ORDERED or QUOTED\_PRICE changes.
* Avoid keeping the Subtotal and instead calculate it dynamically in your queries.
* Use application logic to ensure that the Subtotal is updated as the other values change.

1. How many parts were ordered in order # 21611? (**concept: aggregate data**)

A computer screen with a message

AI-generated content may be incorrect.

To find how many parts were ordered in order #21611, add the amount ordered across all the lines in the order. Because each row in the Order\_Line table corresponds to a specific part of an order, the QTY\_ORDERED column indicates how many of each part was ordered.

1. How much must be paid in total for order 21606? (**concept: aggregate data**)

Answer:

A screenshot of a computer

AI-generated content may be incorrect.

To calculate the total amount due for order #21606, add the Subtotal values for each line item in the order. Each line in the Order\_Line database represents a part, and the Subtotal column gives the price (quantity ordered multiplied by quoted price).

1. To which field in the Customer table should the total amount of an order be added? (**concept: derived data**)

Answer: The total amount of an order is derived data, and adding it to the Customer database is generally discouraged owing to potential redundancy and data integrity issues.

Instead, it is better to:

Calculate it dynamically by querying the Order\_Line table. If you need to store the value for performance reasons, create a separate summary table and update it as needed.

1. What should happen to the Customer CUST\_BALANCE if the customer cancels the order? (**concept: derived data, data consistency**)

Answer: When a customer cancels an order, the CUST\_BALANCE should be reduced by the full value of the canceled order. This change assures data integrity and maintains the customer's balance accuracy.

You have two options for implementing this:

If an order is canceled, manually update the balance.

Automate the updating by setting triggers that alter the balance whenever an order is recorded as canceled. This maintains the system consistent and guarantees that the customer's balance accurately reflects their actual commitments.

1. Who gets the commission for order 21619? How did you determine this? (**concept: foreign key**)

Answer: Based on the REP\_NUM of the customer associated with the order, we may conclude that Salesrep #35 (Richard Hull) will receive the commission for order #21619.

How We Determined This:

We first determined the CUST\_NUM for order #21619, which is 282 (Brookings Direct).

We next looked up CUST\_NUM in the Customer table and discovered that it corresponds to REP\_NUM 35 (Salesrep #35).

Finally, we searched for Salesrep #35 in the Salesrep table to get the commission details.

So, the commission for this order will go to Salesrep #35 (Richard Hull).

1. Calculate how much commission Richard Hull should earn based on the orders he placed. Explain how you calculated this amount. Compare it to the commission the system generated.

Answer: According to the Salesrep table, Richard Hull's commission rate is 8% (0.08). Richard Hull (Salesrep #35) is related to multiple customers via the Customer table. His REP\_NUM = 35 corresponds to the customer numbers 282,408,462 and 608.

A screenshot of a computer

AI-generated content may be incorrect.

This query estimates the total sales (subtotal) for all orders associated with Salesrep #35 (Richard Hull). The JOIN actions integrate data from the Order\_Line, Orders, and Customer tables.

Calculate the Total Sales (Subtotal) for these clients based on the orders they placed. We discovered the following subtotals:

Total sales (subtotal): $6282.80

A computer screen shot of a program

AI-generated content may be incorrect.

This query calculates Richard Hull's commission by multiplying Total Sales by the 8% commission rate (0.08).

A screenshot of a computer

AI-generated content may be incorrect.

To compare the calculated commission to the **system-generated commission**, we query the **Salesrep** table, and the outcome is the same.

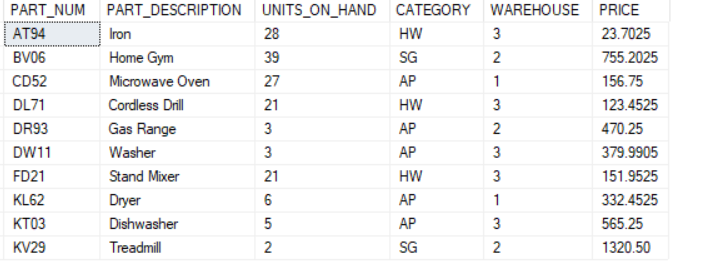
1. Would you be able to determine which customer ordered DR93 if the Order\_Line table also contained the following record? Why not? What must be true about about a foreign key?



**Answer: No, you cannot tell which customer ordered DR93 using only the Order\_Line table since the foreign key link between Order\_Line, Orders, and Customer is absent.**

**To trace the customer, connect the Order\_Line table to the Orders table using ORDER\_NUM, and the Orders table to the Customer table using CUST\_NUM. To ensure data integrity and trace orders back to customers, a foreign key (in this case, ORDER\_NUM and CUST\_NUM) must be correctly constructed and connected between the tables.**

**Part table**



Questions: Study the Part, Order\_Line, Orders, Customer and Salesrep tables

1. How many Treadmills does the company currently have in stock?

Answer:

A screenshot of a computer

AI-generated content may be incorrect.

SELECT UNITS\_ON\_HAND: This selects the column that contains the number of units on hand.

FROM Part: This specifies which table you're querying.

WHERE PART\_DESCRIPTION IS 'Treadmill': This filters the rows to only show the part(s) with the description Treadmill.

1. Can a customer order 4 Treadmills at this time? What should happen if a Customer attempts to order 4 treadmills? (**concept: business rules, data consistency**)

Answer: A consumer cannot order four treadmills at this moment because there are only two available in stock. The system should prohibit such an order by assessing stock levels and either refusing it or offering the option of a backorder or an alternate product.

1. Can I enter any two letters in the Category column? Would it be better if we restrict the entries to HW (HouseWares), SG (Sporting Goods), AP (Appliances) (**concept: domain constraint**)

Answer: It is a good idea to limit entries in the Category column to a subset of valid values, such as HW, SG, and AP, to maintain data integrity and adhere to the domain constraint principle. This would prevent problems like invalid category entries and make the system more stable and manageable.

1. Does it make sense for two different parts to have the same part number? (**concept: primary key**)

Answer: No, it makes no sense for two distinct parts to share the same part number. To ensure data integrity and compliance with primary key constraints, the component number should be unique to each part. This ensures that the system can correctly identify and manage every element individually.

1. Is it possible for two different parts to have the same or similar part description?

Answer: Two components can have identical or similar descriptions; however, this is undesirable and can lead to confusion, errors, and reporting concerns. To preserve clarity and minimize business problems, it is better to guarantee that each part has a unique, unambiguous description and that part numbers are used as the unique identifiers for the parts.

1. Is it possible for two different parts to be stored in the same warehouse? Is it considered to be redundancy if warehouse 1 appears in more than one row (record) within the Part table? Why not? Is it redundant if the same part number appears in two different rows (records) within the part table (**concept: redundancy)**

**Answer:** Yes, it is possible to keep two separate pieces in the same warehouse. The warehouse column in the Part table specifies where the part is physically situated, and it is normal for several parts to be placed in the same warehouse if that warehouse houses a variety of products.

No, having warehouse 1 appear in more than one row in the Part table is not considered redundancy. This is not redundant because each part is still uniquely identified by its part number (which should be a primary key), and the warehouse location is simply one of the parts' attributes.

Yes, it is redundant to have the same part number appear in two different rows (records) of the Part table for the same part.

The part number should be each part's unique identification, appearing only once in the table.

If the same part number appears in numerous rows, it causes data redundancy, which occurs when the same information (such as description and price) is stored more than once.

1. Has the company sold one or more units of every part that the company stores in one of its warehouses? How did you determine this information ? (**concept: foreign key**)

Answer:

A computer screen shot of a computer

AI-generated content may be incorrect.

To see if the company has sold one or more units of each part held in its warehouse, we compare the Part table (which represents what the company stores) to the Order\_Line table (which represents what has been sold).

If this query produces no results, it signifies that each part has been sold at least once.

If it returns any rows, the parts have not yet been sold.

Despite the fact that the DL71 (Cordless Drill) is stored at one of the company's warehouses, no units have been sold.

1. Should we be allowed to enter the following record into the Order\_Line table? Why? (**concept: foreign key**)



Answer: No, you should not be able to add this record to the Order\_Line database because "DR88" does not exist in the Part table.

This violates the foreign key requirement, which states that only legitimate, existent components may be ordered.

1. How many different parts are stored in each warehouse? (**concept: aggregate function**)

A screenshot of a computer

AI-generated content may be incorrect.

This query counts the number of unique components (PART\_NUM) in each warehouse and organizes the results by WAREHOUSE.

1. How many different parts belong to each category? (**concept: aggregate function**)

Answer:

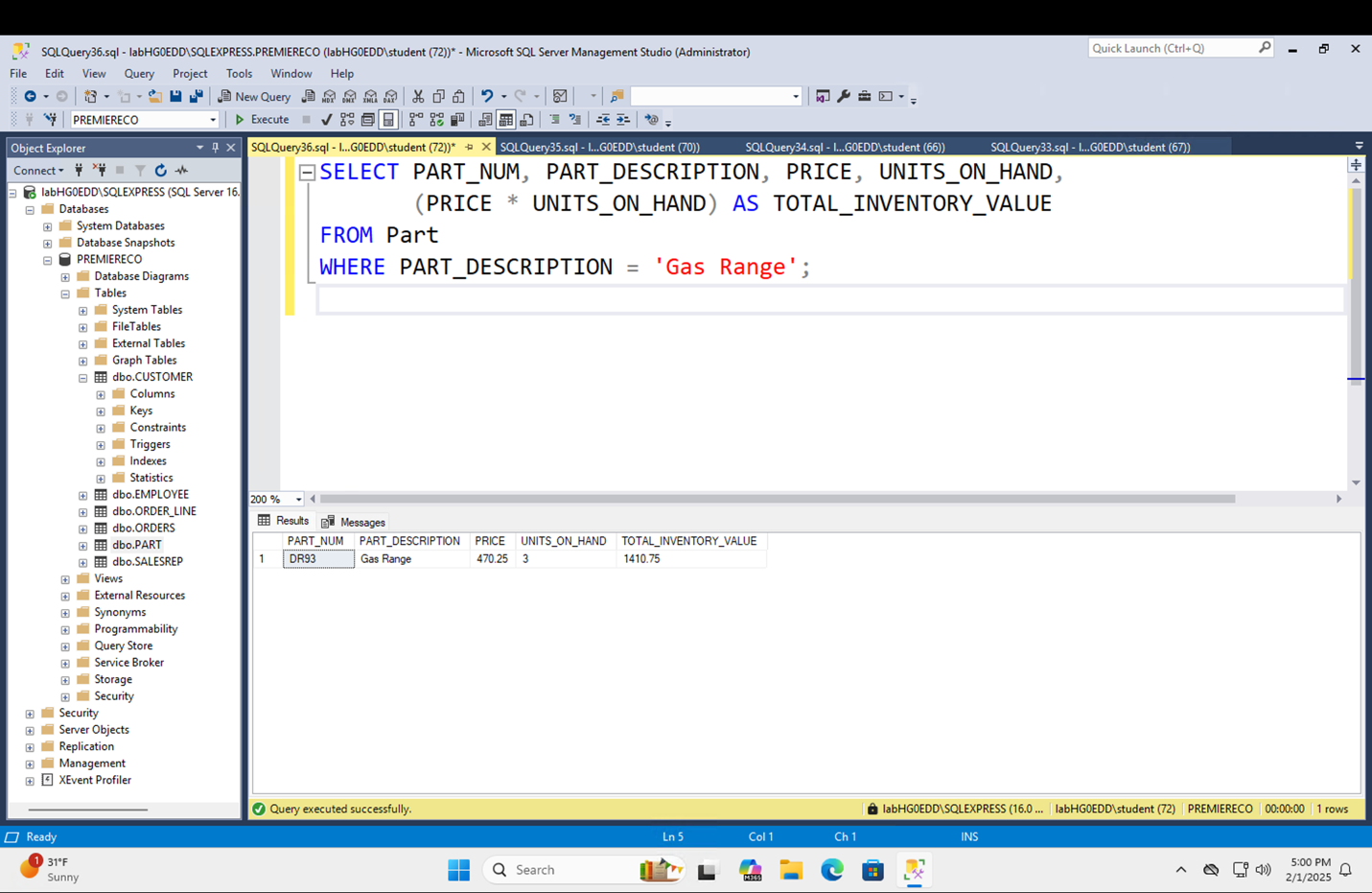
A screenshot of a computer

AI-generated content may be incorrect.

This query provides the number of distinct parts in each category (such as HW, SG, and AP), allowing you to determine how many different types of parts belong to each category.

1. How much is the total inventory of Gas Ranges worth? (**concept: calculated column**). Why do think this information is not stored in the Part table?

Answer:



The overall inventory value of Gas Ranges is best determined on the fly by combining pricing and quantity data. Storing it directly in the Part table would cause redundancy and data consistency issues.

The Part database does not hold the total inventory value of parts (such as Gas Ranges) because:

Dynamic Calculation: The inventory value is determined by both price and quantity, which might fluctuate over time. Storing the value would necessitate frequent updates to ensure correctness.

Data Integrity: Storing the value would result in redundancy and inconsistency if price or quantity changes were not reflected in the inventory value.

Normalization: To reduce redundancy and ensure data consistency, the database should hold raw data (price and quantity) together with calculated values as needed.

1. List the name of each Customer who has ordered an Iron . (**concept: foreign key**)

Answer:

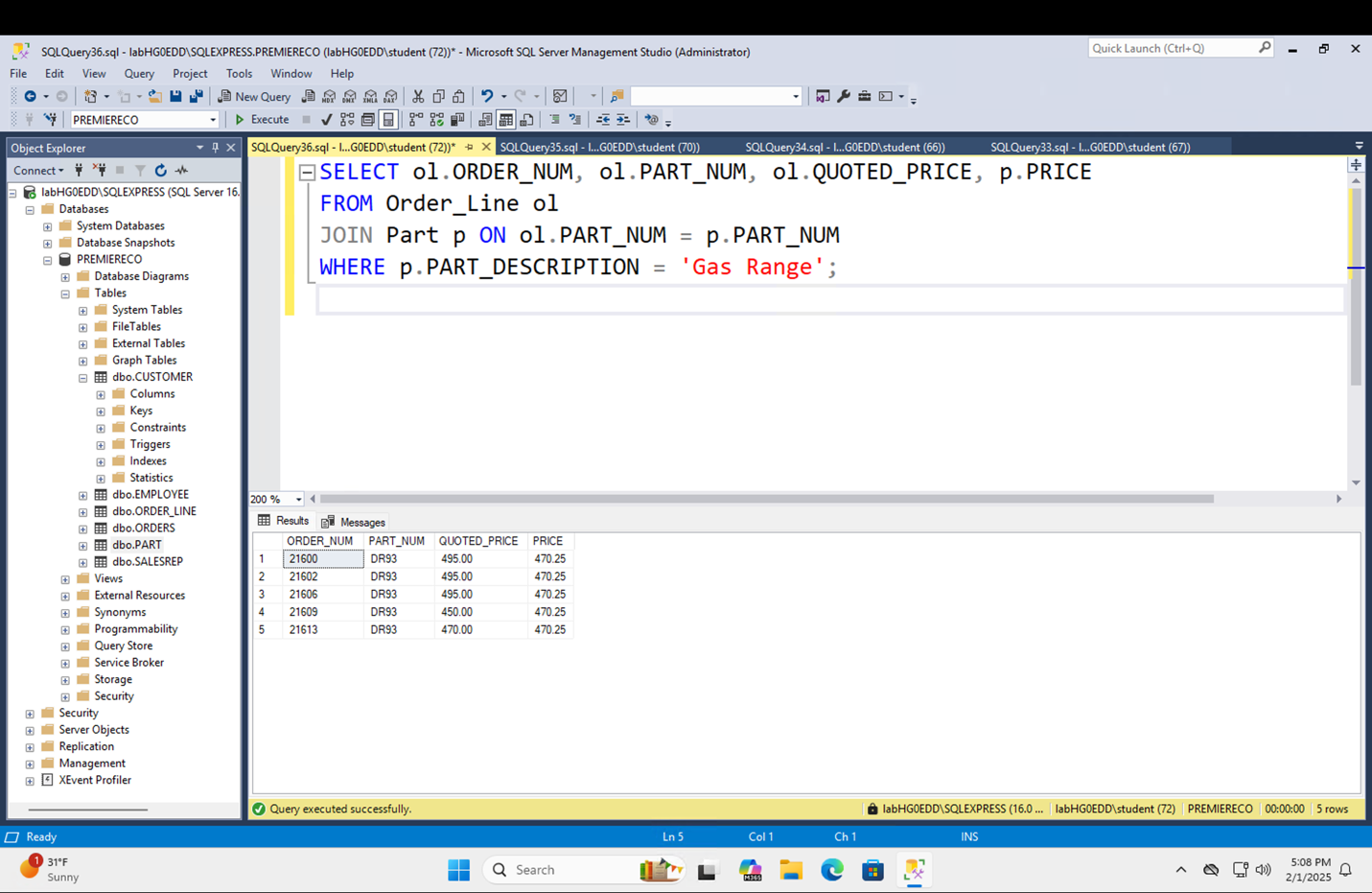
A computer screen shot of a computer screen

AI-generated content may be incorrect.

Based on the tables' foreign key associations, this query returns a list of customer names who have ordered an Iron (part AT94).

1. Has the Gas Range been sold for the same price as it is listed in the Part table? Is this an inconsistency? Why might it not be an inconsistency?

Answer:



The Gas Range (part DR93) has not been offered at the price mentioned in the Part Table.

The reported price in the Part table for the Gas Range is 470.25.

The quoted price (what buyers paid) in the Order\_Line table fluctuates:

For some orders, the advertised price is 495.00.

Other orders are quoted at $450.00 or $470.00.

No, this isn't an inconsistency. The variation between the quoted and displayed pricing could be due to discounts, promotions, or negotiated prices for special orders. The Part table reflects the standard price (470.25), but the Order\_Line table indicates the actual selling price (which may differ from the published price).

1. If we place the following order (it appears on the next page), what records should be **inserted** and/or **modified** in the Customer, Part and SalesRep tables?



Answer: The customer Table: Make no modifications to client information unless they are new customers.

Part Table: Change the Units\_On\_Hand for DW11 to reflect the drop in stock.

SalesRep Table: No changes will be made until a new link between a sales rep and the customer is needed. However, the sales representative's compensation should be tracked about the order.

1. If we wish to place the following order what records must be **inserted** into the database?



Answer: Create a new entry in the Order Table and link it to the client.

Create an Order Line Table to record the ordered items, including part number, quantity, price, and subtotal.

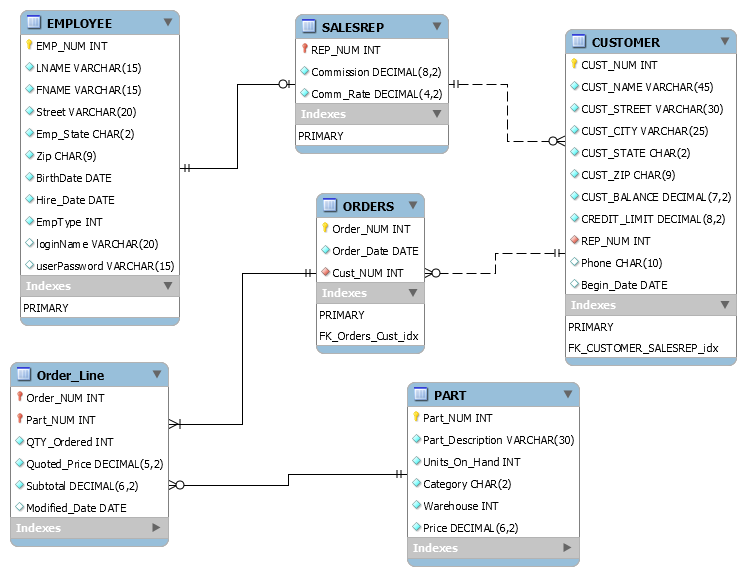
Update the part's (DW11) inventory to match the specified quantity.

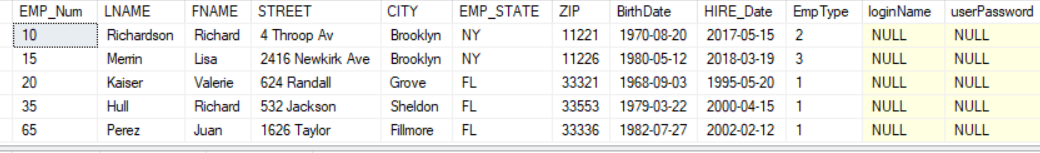
Customer Table: No adjustments are required until a new customer places an order.

If a sales representative is involved in order processing, add or update records in the SalesRep Table (optional).

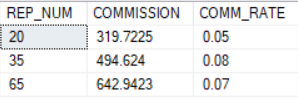
Design View of the PremiereCo database ERD

* Each rectangle represents one entity
* A primary key must be designated for each entity
* In some cases, foreign keys must be designated to link tables
* Tables can be linked in three manners: 1:1, 1:M, M:N
* Some relationships are mandatory, others are optional

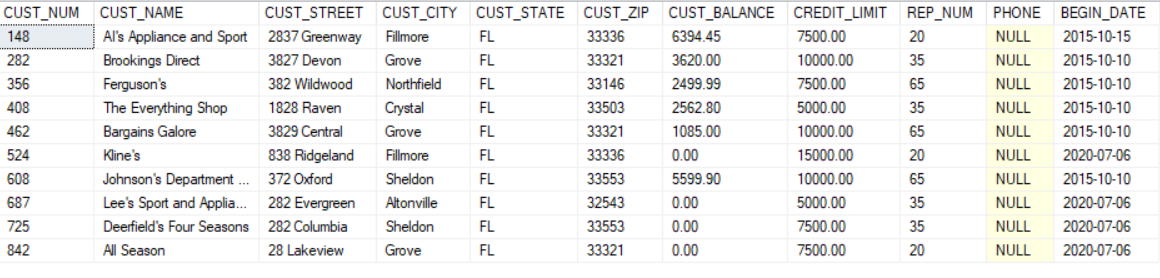


PremiereCo

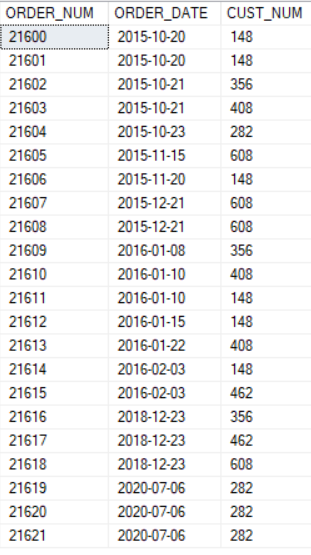
Employee



Salesrep

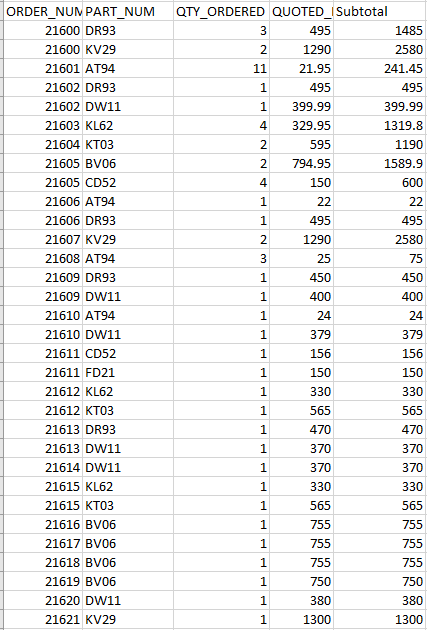


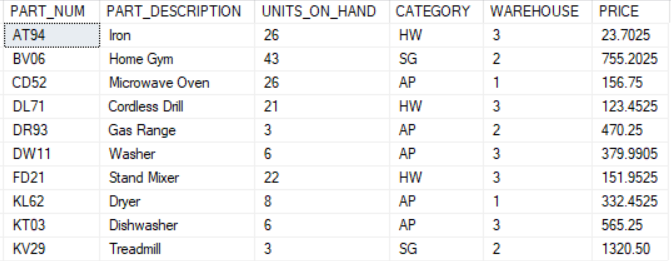
Customer



Order\_Line

Orders





Part