Super Store Database

MYSQL Project - 1

PRESENTED BY:

Jeevarathnam R T

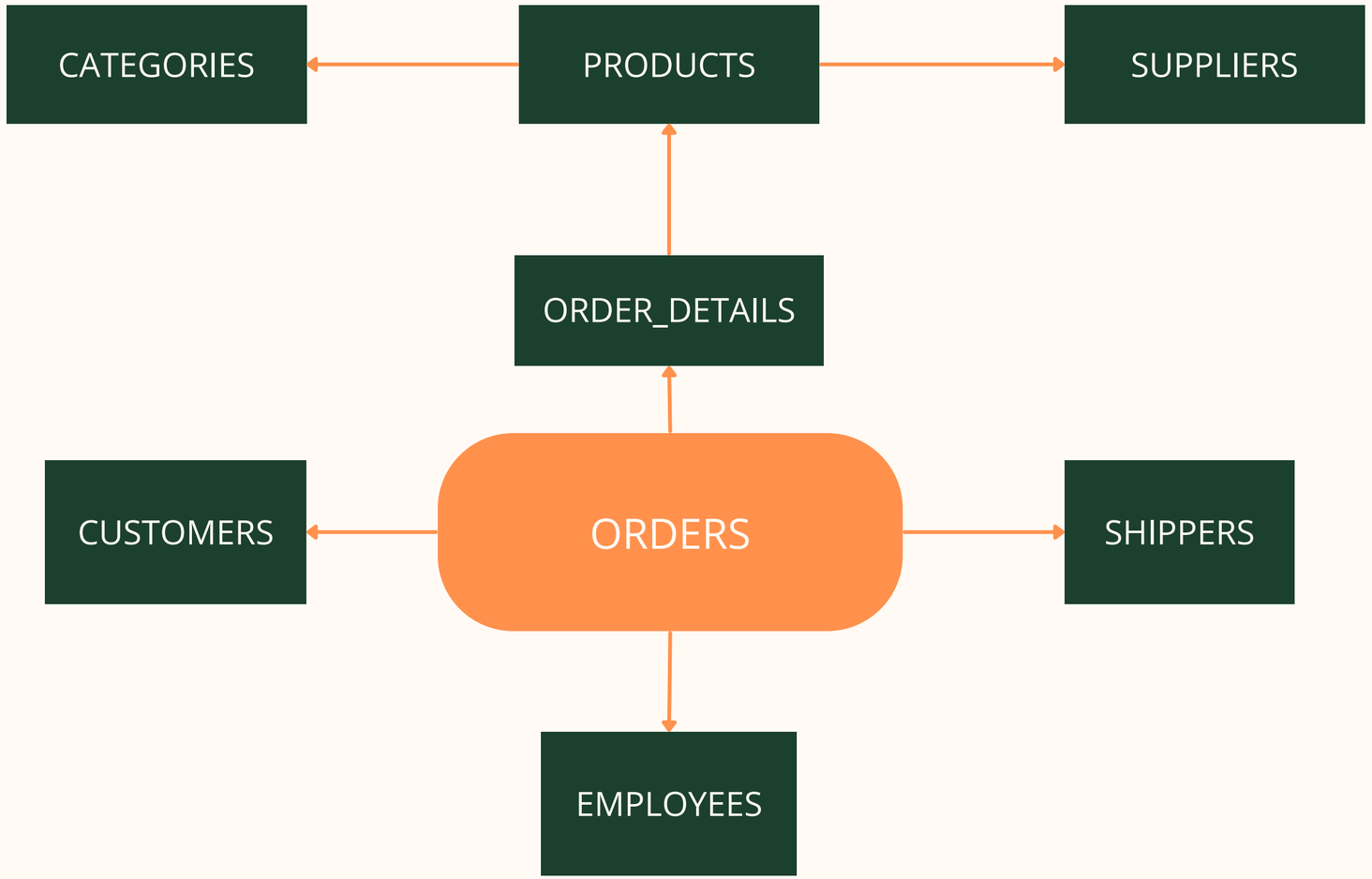
Agenda

In the upcoming slide You will see
The Function which are
Mentioned below:

- Data Retrieval and Basic Queries
- Filtering and Sorting Data
- Joining Tables
- Data Aggregation and Grouping
- Advanced SQL Functions
- Creating Views and Stored Procedures
- Apply subqueries and common table expressions (CTEs)
- Data Insights and Reporting

Database structure

Back to Agenda



Basic Queries

Back to Agenda

```sql
-- Basic Queries
-- What are the names of all customers from the Customers table?
select distinct Customername, Country
FROM customers;

-- Retrieve all products and their prices from the Products table.
SELECT ProductName , Price
FROM products;
```

Customername	Country
Alfreds Futterkiste	Germany
Ana Trujillo Emparedados y helados	Mexico
Antonio Moreno Taquería	Mexico
Around the Horn	UK
Berglunds snabbköp	Sweden
Blauer See Delikatessen	Germany

ProductName	Price
Chais	18
Chang	19
Aniseed Syrup	10
Chef Anton's Cajun Seasoning	22
Chef Anton's Gumbo Mix	21.35
Grandma's Boysenberry Spread	25
Unde Bob's Organic Dried Pears	30
Northwoods Cranberry Sauce	40

- Extract information from individual tables to understand the structure and contents of the database.
- Retrieve lists of customers, products, categories, employees, shippers, and orders using simple SELECT queries.

Filtering Data

Back to Agenda

```sql
-- Filtering Data
-- List all customers located in the 'USA' from the Customers table.
SELECT CustomerName, Country
FROM Customers
WHERE country = 'USA';

-- Find all employees who were born after January 1, 1960, from the Employees table.
SELECT EmployeeID, concat_ws(" ",Firstname,lastname) AS EmployeeName, Birthdate
FROM employees
WHERE year(Birthdate) >= 1960;
```

CustomerName	Country
Great Lakes Food Market	USA
Hungry Coyote Import Store	USA
Lazy K Kountry Store	USA
Let's Stop N Shop	USA
Lonesome Pine Restaurant	USA
Old World Delicatessen	USA
Rattlesnake Canyon Grocery	USA
Save-a-lot Markets	USA

EmployeeID	EmployeeName	Birthdate
1	Nancy Davolio	1968-12-08
3	Janet Leverling	1963-08-30
6	Michael Suyama	1963-07-02
7	Robert King	1960-05-29
9	Anne Dodsworth	1969-07-02

- Implement filtering to retrieve specific subsets of data, such as customers from a particular country or employees born after a certain date.
- Sort data based on various attributes, such as product prices, order dates, or customer names.

Joins

Back to Agenda

```sql
-- Joins
-- Retrieve a list of all orders along with the customer names and the shipper names.
SELECT o.OrderID, c.CustomerName AS "Customer's Name", s.shippername AS "Shipper's Name"
FROM orders o JOIN customers c ON o.customerid = c.customerid
JOIN shippers s ON o.shipperid = s.shipperid;

-- Get a list of all products along with their category names.
SELECT p.ProductName, c.categoryName
FROM Products p JOIN Categories c ON p.categoryID = c.categoryID
ORDER BY c.categoryName;
```

	OrderID	Customer's Name	Shipper's Name
	10440	Save-a-lot Markets	United Package
	10441	Old World Delicatessen	United Package
	10442	Ernst Handel	United Package
	10248	Wilman Kala	Federal Shipping
	10255	Richter Supermarkt	Federal Shipping
	10257	HILARIÓN-Abastos	Federal Shipping

	ProductName	categoryName
	Pâté chinois	Meat/Poultry
	Uncle Bob's Organic Dried ...	Produce
	Tofu	Produce
	Rössle Sauerkraut	Produce
	Manjimup Dried Apples	Produce
	Longlife Tofu	Produce
	Ikura	Seafood

- Perform inner joins, left joins, right joins, and full joins to combine data from multiple tables and retrieve comprehensive information, such as order details along with customer and employee names.
- Join tables to analyze relationships between entities, such as products and their categories or orders and their shippers.

Aggregating and Grouping

Back to Agenda

```sql
-- Aggregation and Grouping
-- Find the total number of orders placed by each customer.
SELECT c.CustomerID, c.CustomerName, Count(o.orderid) AS "NO. of orders placed"
FROM customers c JOIN orders o ON c.customerid = o.customerid
GROUP BY c.customerid
ORDER BY Count(o.orderid) DESC;

-- Calculate the total quantity of products ordered for each product.
SELECT  p.ProductName,  count(od.orderid) AS "Total Quantity Ordered"
FROM Products p JOIN order_details od ON p.productid = od.productid
GROUP BY p.ProductName
ORDER BY Count(od.orderid) DESC;
```

CustomerID	CustomerName	NO. of orders placed
20	Ernst Handel	10
63	QUICK-Stop	7
87	Wartian Herkku	7
65	Rattlesnake Canyon Grocery	7
75	Split Rail Beer & Ale	6
37	Hungry Owl All-Night Grocers	6

ProductName	Total Quantity Ordered
Raclette Courdavault	14
Gorgonzola Telino	14
Mozzarella di Giovanni	14
Fløtemysost	13
Tarte au sucre	13
Gnocchi di nonna Alice	12

- Use aggregate functions like COUNT, SUM, AVG, MAX, and MIN to generate summaries of the data, such as total orders per customer or average product price per category.
- Group data by specific attributes to generate meaningful reports, such as the number of orders per shipper or total sales per product category.

Complex queries

Back to Agenda

```sql
-- Complex Queries
-- List all employees who have handled orders in the Month of July 1996.
SELECT e.employeeid, concat_ws(" ",e.Firstname,e.lastname) AS EmployeeName,
       count(o.orderid) AS "No. of orders handled"
FROM employees e
JOIN orders o ON e.employeeid = o.employeeid
WHERE year(orderdate) = 1996 AND month(orderdate) = 7
GROUP BY e.employeeid
ORDER BY count(o.orderid) DESC;

-- Find the most popular product category based on the number of products ordered.
SELECT c.CategoryName, Count(o.orderID) AS "Total Count",
       DENSE_RANK() OVER( ORDER BY Count(o.orderID) DESC) AS "Rank"
FROM categories c JOIN Products p
     ON c.categoryid = p.categoryid
JOIN order_details o ON p.productid = o.productid
GROUP BY c.CategoryName
ORDER BY Count(o.orderID) DESC;
```

employeeid	EmployeeName	No. of orders handled
4	Margaret Peacock	7
3	Janet Leverling	4
5	Steven Buchanan	3
6	Michael Suyama	2
8	Laura Callahan	2
9	Anne Dodsworth	2
1	Nancy Davolio	1
2	Andrew Fuller	1

CategoryName	Total Count	Rank
Dairy Products	100	1
Beverages	93	2
Confections	84	3
Seafood	67	4
Meat/Poultry	50	5
Condiments	49	6
Grains/Cereals	42	7
Produce	33	8

- Here we used different function used in before slide altogether at one query
- Optimize query to retrieve the most relevant data

Create a VIEW

Back to Agenda

```sql
/*Create a view with order price and other details needed together*/
Create View order_price AS (
                    SELECT od.orderID,o.customerid, od.Quantity * p.Price AS Order_price, o.shipperid, o.employeeid
                    FROM order_details od
                    JOIN products p ON od.productid = p.productid
                    JOIN orders o ON od.orderid = o.orderid
                    GROUP BY od.orderid
);
```

- We use view to create a temporary table to retrieve data from.
- CTE’s are used with the query .
- View can be called from any query once created.
- View can be created within single table with aggregation or combination of two or more tables .
- It increases optimized timing and simplify with reusability.
- Here we need total amount of order .
- So we create a view with aggregated column of quantity and price for sum of price to get the order price

Subqueries

```sql
SELECT ProductName
FROM products
WHERE productID= (SELECT ProductID
                  FROM products
                  ORDER BY price ASC
                  limit 1);

/*Select shipper together with the total price of proceed orders*/
with pricing AS ( SELECT od.orderID,od.Quantity * p.Price AS Order_price, o.shipperid
                  FROM order_details od
                  JOIN products p ON od.productid = p.productid
                  JOIN orders o ON od.orderid = o.orderid
                  GROUP BY od.orderid)
Select s.Shippername, sum(p.order_price) AS Total_order_price
from Pricing p
JOIN shippers s ON p.shipperid = s.shipperid
group by s.shippername
```

Productname
Chais
Chang
Aniseed Syrup
Chef Anton's Cajun Seasoning
Chef Anton's Gumbo Mix
Grandma's Boysenberry Spread
Uncle Bob's Organic Dried Pears
Northwoods Cranberry Sauce

Back to Agenda

Shippername	Total_order_pric
Federal Shipping	52273.11000000
United Package	46582.2
Speedy Express	29690.27

- Use subqueries to combine and compare data across multiple tables.
- Break down complex queries into simpler subqueries to improve readability and maintainability.

Subqueries & CTE’s

Back to Agenda

```sql
/*Select name of the cheapest product (only name) using subquery*/
SELECT ProductName
FROM products
WHERE productID= (SELECT ProductID
                  FROM products
                  ORDER BY price ASC
                  limit 1);

/*Select shipper together with the total price of proceed orders*/
with pricing AS ( SELECT od.orderID,od.Quantity * p.Price AS Order_price, o.shipperid
                  FROM order_details od
                  JOIN products p ON od.productid = p.productid
                  JOIN orders o ON od.orderid = o.orderid
                  GROUP BY od.orderid)
Select s.Shippername, sum(p.order_price) AS Total_order_price
from Pricing p
JOIN shippers s ON p.shipperid = s.shipperid
group by s.shippername
order by sum(p.order_price) DESC;
```

- Subqueries and Common Table Expressions (CTEs) to break down complex queries and improve readability and maintainability.
- Optimize query performance by reducing data retrieval complexity.
- Enable dynamic and conditional data retrieval.
- They simplify complex queries by breaking them into manageable parts.

Data Insights and Reporting

Back to Agenda

```sql
/*Select Total orders for each shipping company*/
SELECT s.ShipperName, Count(o.OrderID) AS "No. of Orders"
FROM orders o
JOIN shippers s ON o.shipperid = s.shipperid
GROUP BY  s.ShipperName
ORDER BY Count(o.OrderID) DESC;
```

ShipperName	No. of Orders
United Package	74
Federal Shipping	68
Speedy Express	54

- United packages has handled the most no of shipments handled
- The sum of total no of orders handled is 74

Back to Agenda

```sql
/*Select customer who spend the  most money top 2*/
select  op.customerid,c.contactname, sum(op.order_price) AS Purchase_price
from order_price op
JOIN Customers c ON op.customerid = c.customerid
GROUP BY op.customerid
ORDER BY Purchase_price DESC
LIMIT 2;
```

	customerid	contactname	Purchase_price
	73	Jytte Petersen	13195
	20	Roland Mendel	12041.65

The top 2 customers who have contributed the company revenue are :

- Jytte petersen
- Roland mendel

Back to Agenda

```sql
/*select country and their order price*/
SELECT c.Country , Sum(o.order_price) AS Order_price_by_country
FROM order_price o
JOIN customers c ON o.customerID = c.CustomerID
GROUP BY C.country
ORDER BY Order_price_by_country DESC;
```

Country	Order_price_by_country
USA	17094.219999999998
Germany	16772.9
Austria	14907.4
Denmark	14449
Argentina	279
Norway	67.5

Top contries by purchase power are

1. USA
2. Germany
3. Austria
4. Denmark

Countries we need to develop on:

- Argentina
- Norway

Back to Agenda

```sql
/*select ordered price for each month*/
SELECT month(o.OrderDate),
       CAST(SUM(op.order_price) AS DECIMAL(10, 2)) AS Total_Spending,
       RANK() OVER (ORDER BY SUM(op.order_price)  DESC) AS SpendingRank
FROM order_price op
JOIN orders o ON op.orderID = o.orderID
group by month(OrderDate)
ORDER BY month(OrderDate);
```

month(o.OrderDate)	Total_Spending	SpendingRank
1	34246.62	1
2	6446.00	8
7	13517.25	4
8	12727.74	6
9	11363.80	7
10	13393.20	5
11	19562.50	2
12	17288.47	3

- Top selling month is **Jan**
- November and December has high sales compared to other months
- we could see a **Trend** on last two months of the year

Back to Agenda

```sql
/*select supplier with highest number of orders
processed and total amount of order per supplier*/
SELECT s.SupplierName,s.Country,
       count(op.orderID) AS "No. of orders processed",
       CAST(SUM(op.order_price) AS DECIMAL(10, 2)) AS Total_earnings
FROM Suppliers s
JOIN products p ON s.supplierID = p.supplierID
JOIN order_details od ON p.productid = od.productid
JOIN order_price op ON od.orderID = op.orderID
group by s.SupplierName
order by Total_earnings desc;
```

SupplierName	Country	No. of orders processed	Total_earnings
Plutzer Lebensmittelgroßmärkte AG	Germany	42	39073.34
Specialty Biscuits, Ltd.	UK	33	34688.10
Pavlova, Ltd.	Australia	40	29861.57
Aux joyeux ecclésiastiques	France	15	28300.50

- These suplpliers have contributed more to the store than other suppliers
- Total earnings from these suppliers to the country can say that these suppliers have high productivity

Back to Agenda

```sql
/*select employees and their revenue generated for company */
select e.EmployeeID, concat_ws(" ",e.firstname,e.lastname) AS Employee_name ,
       CAST(SUM(op.order_price) AS DECIMAL(10, 2)) AS Employee_revenue,
       Dense_rank() OVER(ORDER BY SUM(op.order_price) DESC) AS "Rank by revenue"
FROM Employees e
JOIN orders o ON e.employeeID = o.employeeID
JOIN order_price op ON o.orderID = op.orderID
GROUP BY e.EmployeeID;
```

EmployeeID	Employee_name	Employee_revenue	Rank by revenue
4	Margaret Peacock	40697.34	1
1	Nancy Davolio	23479.51	2
3	Janet Leverling	20604.05	3
8	Laura Callahan	14377.17	4
2	Andrew Fuller	11601.06	5

- Marget peacock has made a high revenue
- second person have revenue of half of marget

Back to Agenda

```sql
/*Find the total revenue of Super store*/
SELECT CAST(SUM(order_price) AS DECIMAL(10, 2)) AS Total_revenue
FROM order_price;
```

Total_revenue
128545.58

Total revenue of store is **128,545$**

Back to Agenda

- These are the examples to show how we can use MYSQL to retrieve data from Database.
- Insights which retrieved shows the trends and values from data
- Trends - Nov, Dec has high sells value
- Values - Highest revenue employee, highest sold suppliers etc.
- We can further dive into the data and retrieve most important info to help our client
- We can get information's from original data to support our assumptions or prove it.

Recommendations

Back to Agenda

Use simple and clear queries

Avoid complexing queries and try to optimize it

Use views , CTE’s to simplify

Learn and practice SQL and experiment in it.

Get In Touch

Email

Jeevarathinam969@gmail.com

Social Media

@jeevarathnam R T