==========

Pizza Sales Management System
Optimizing Pizza Sales and Inventory Management

Prepared By: Ravi Mishra

Institute- IT. VEDANT

Date: November 11, 2024

==========

Project Overview:
This project aims to develop a comprehensive Pizza Sales Management System to streamline and optimize pizza sales, inventory tracking, and customer management. The system utilizes SQL for database management, ensuring data integrity and easy access to information, ultimately enhancing operational efficiency and customer satisfaction.

➤Creating a database

Database can be created using below query where you can store tables

Query 1 | SQL File 1* | SQL File 3*

Limit to 1000 rows

```
-- for creating new database in mysql.

create database my_collage;
```

➤ Creating a table using constraints

Table is created to store data in it and we should assign datatype to each column in table. Constraint is a rule applied to column in a table to ensure the accuracy

Query 1 | SQL File 1* | SQL File 3*

Limit to 1000 rows

```sql
-- for creating new table in database.
use my_collage;
CREATE TABLE students_records (
    First_name VARCHAR(50),
    Last_name VARCHAR(50),
    roll_no INT PRIMARY KEY NOT NULL,
    mobile_no BIGINT UNIQUE,
    date_of_birth DATE,
    city VARCHAR(50),
    state VARCHAR(76)
);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

First_name	Last_name	roll_no	mobile_no	date_of_birth	city	state
NULL	NULL	NULL	NULL	NULL	NULL	NULL

➤Inserting value in table

For inserting data in table, apply the below query and then add values in the column.

```
-- for adding values in table.

insert into students_records
values
("Rahul","Mishra",01, 8112453626,"1999-10-19","mumbai","Maharastra"),
("Vikash","Sharma",02,7894564562,"1999-02-20","kurla","Panjab"),
("Sandeep","Verma",03,8877562565,"1998-10-19","Thane","Dhelhi"),
("Vijay","Chavan",04,5469895656,"1999-12-25","Kalyan","Uttar_pradesh"),
("Akash","Mali",05,8798656585,"1999-05-28","Dadar","Bhopal");
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

First_name	Last_name	roll_no	mobile_no	date_of_birth	city	state
Rahul	Mishra	1	8112453626	1999-10-19	mumbai	Maharastra
Vikash	Sharma	2	7894564562	1999-02-20	kurla	Panjab
Sandeep	Verma	3	8877562565	1998-10-19	Thane	Dhelhi
Vijay	Chavan	4	5469895656	1999-12-25	Kalyan	Uttar_pradesh
Akash	Mali	5	8798656585	1999-05-28	Dadar	Bhopal
NULL	NULL	NULL	NULL	NULL	NULL	NULL

➤Renaming a column in a table:

For renaming column name first we have to apply alter table command then apply rename query.

Query 1 | SQL File 1* | SQL File 3*

Limit to 1000 rows

```
-- for Rename Column

alter table students_records
rename column city to Birth_place;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

First_name	Last_name	roll_no	mobile_no	date_of_birth	Birth_place	state
Sandeep	Verma	3	8877562565	1998-10-19	Thane	Dhelhi
Rahul	Mishra	1	8112453626	1999-10-19	mumbai	Maharastra
Vikash	Sharma	2	7894564562	1999-02-20	kurla	Panjab
Vijay	Chavan	4	5469895656	1999-12-25	Kalyan	Uttar_pradesh
Akash	Mali	5	8798656585	1999-05-28	Dadar	Bhopal
NULL	NULL	NULL	NULL	NULL	NULL	NULL

➤Drop and truncate table:

Drop table means whole table will be dropped including data as well as table structure and truncate table means only data will be deleted but the structure will remain same

```
-- drop And truncate Column

drop table students_records;
truncate table students_records;
```

➤Delete command:

Delete command is used for deleting records based on specific conditions.

```
-- delete command

delete from students_records
where First_name ="Vijay";
```

➤Update function:

Update is used for updating records in the table, example of update query is shown below

```
3 •    set sql_safe_updates=0;
4
5 •   update my_collage.students_records
6     set state = "Delhi" where state="Dhelhi";
7
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

First_name	Last_name	roll_no	mobile_no	date_of_birth	Birth_place	state
Rahul	Mishra	1	8112453626	1999-10-19	mumbai	Maharastra
Vikash	Sharma	2	7894564562	1999-02-20	kurla	Panjab
Sandeep	Verma	3	8877562565	1998-10-19	Thane	Delhi
Vijay	Chavan	4	5469895656	1999-12-25	Kalyan	Uttar_pradesh
Akash	Mali	5	8798656585	1999-05-28	Dadar	Bhopal
NULL	NULL	NULL	NULL	NULL	NULL	NULL

➤Comparison and logical operator

Comparison and logical operators are used for comparing the values and using logical functions respectively.

```
select pizza_id,pizza_type_id,size,price
from pizzas
where price between 16 and 20;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_id	pizza_type_id	size	price
bbq_ckn_m	bbq_ckn	M	16.75
cali_ckn_m	cali_ckn	M	16.75
ckn_alfredo_m	ckn_alfredo	M	16.75
ckn_pesto_m	ckn_pesto	M	16.75
southw_ckn_m	southw_ckn	M	16.75

```
7  select *from pizzas
8  where price >16.50
9  and pizza_type_id ="bbq_ckn"
10 and size ="M";
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_id	pizza_type_id	size	price
bbq_ckn_m	bbq_ckn	M	16.75

➤Arithmetic operators:

Arithmetic operators are used to find various operations such as sum, max, min, avg, etc

```
SELECT
    SUM(price) AS sum_of_total_price,
    AVG(price) AS avg_price,
    MIN(price) AS min_price,
    MAX(price) AS max_price
FROM
    pizzas;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sum_of_total_price	avg_price	min_price	max_price
1578.3	16.440625	9.75	35.95

➤Math functions

In math function we can do operations such as absolute, floor, ceil, round.

```
21 •    SELECT
22      ABS(- 500) AS whole_on,
23      CEIL(- 5.6) AS ceil_result,
24      FLOOR(- 5.6) AS floor_result,
25      CEIL(5.6) AS ceil_result_1,
26      FLOOR(5.6) AS floor_result_1,
27      ROUND(5.4) AS round_result,
28      ROUND(5.6) AS round_result_1,
29      POWER(2, 5) AS power,
30      SQRT(775) AS sqrt;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

whole_on	ceil_result	floor_result	ceil_result_1	floor_result_1	round_result	round_result_1	power	sqrt
500	-5	-6	6	5	5	6	32	27.83882181415011

➤Like function

It is used to find the word starting with , ending with or containing some letter in it.

```
select pizza_type_id,name
from pizza_types
where pizza_type_id like ("c%")
and  pizza_type_id like ("%o");
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_type_id	name
ckn_alfredo	The Chicken Alfredo Pizza
ckn_pesto	The Chicken Pesto Pizza

The Chicken Pesto Pizza

➤Distinct:

Distinct keyword is used to retrieve unique values from a specified column.

```
9   select distinct pizza_type_id
10  from pizza_types;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_type_id
bbq_ckn
cali_ckn
ckn_alfredo
ckn_pesto
southw_ckn

Resets all sorted columns

➤Case statement:

The case statement gives us result according to given conditions, example of case statement is given below

```
34 •        SELECT
35          price,
36          CASE
37              WHEN price > 16 THEN 'High_price'
38              WHEN price >= 10 THEN 'Midium_price'
39              WHEN price < 10 THEN 'Low'
40          END AS Price_cetogery
41      FROM
42          pizzas;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

price	Price_cetogery
12.75	Midium_price
16.75	High_price
20.75	High_price
12.75	Midium_price
16.75	High_price
20.75	High_price
12.75	Midium_price

➤Order by:

Order by is nothing but arranging the values in the column in Ascending order or descending order.

```
44 •    SELECT
45       pizza_id, pizza_type_id, price
46   FROM
47       pizzas
48   ORDER BY price DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_id	pizza_type_id	price
the_greek_xxl	the_greek	35.95
the_greek_xl	the_greek	25.5
brie_carre_s	brie_carre	23.65
ital_veggie_l	ital_veggie	21
spinach_supr_l	spinach_supr	20.75

```
44 •    SELECT
45       pizza_id, pizza_type_id, price
46   FROM
47       pizzas
48   ORDER BY price asc;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_id	pizza_type_id	price
pepperoni_s	pepperoni	9.75
hawaiian_s	hawaiian	10.5
pep_msh_pep_s	pep_msh_pep	11
four_cheese_s	four_cheese	11.75
ital_cpcllo_s	ital_cpcllo	12
veggie_veg_s	veggie_veg	12

Retrieve the total number of orders placed.

```
-- Retrieve the total number of orders placed.

SELECT
    COUNT(order_id) AS orders_placed
FROM
    orders;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

orders_placed
21350

Joins

In SQL, joins are used to combine rows from two or more tables based on a related column between them.

Here are the most common types of joins:

- Inner Join
- Left Join (Left Outer Join)
- Right Join (Right Outer Join)
- Full Join (Full Outer Join)
- Cross Join
- Self Join

LEFT JOIN

A **LEFT JOIN** in SQL is used to retrieve all records from the left table (the first table in the join statement), and the matched records from the right table (the second table in the join statement).

```
4 •  select * from orders_details
5    left join pizzas
6    on pizzas.pizza_id = orders_details.pizza_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_details_id	order_id	pizza_id	quantity	pizza_id	pizza_type_id	size	price
1	1	hawaiian_m	1	hawaiian_m	hawaiian	M	13.25
2	2	classic_dlx_m	1	classic_dlx_m	classic_dlx	M	16
3	2	five_cheese_l	1	five_cheese_l	five_cheese	L	18.5
4	2	ital_supr_l	1	ital_supr_l	ital_supr	L	20.75
5	2	mexicana_m	1	mexicana_m	mexicana	M	16
6	2	thai_ckn_l	1	thai_ckn_l	thai_ckn	L	20.75

RIGHT JOIN

A **RIGHT JOIN** (or right outer join) in SQL is used to retrieve all records from the right table (the second table in the join statement), and the matched records from the left table (the first table in the join statement).

```
select *from orders_details
right join  pizzas
on pizzas.pizza_id = orders_details.pizza_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_details_id	order_id	pizza_id	quantity	pizza_id	pizza_type_id	size	price
8895	3899	cali_ckn_l	1	cali_ckn_l	cali_ckn	L	20.75
8881	3892	cali_ckn_l	1	cali_ckn_l	cali_ckn	L	20.75
8880	3892	bbq_ckn_m	1	bbq_ckn_m	bbq_ckn	M	16.75
8869	3887	bbq_ckn_s	1	bbq_ckn_s	bbq_ckn	S	12.75
8868	3887	bbq_ckn_m	1	bbq_ckn_m	bbq_ckn	M	16.75
8861	3886	cali_ckn_l	2	cali_ckn_l	cali_ckn	L	20.75

Cross Join

It Returns the Cartesian product of both tables, meaning it combines each row of the first table with each row of the second table.

```
select  *from orders_details
cross join pizzas
on pizzas.pizza_id = orders_details.pizza_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

order_details_id	order_id	pizza_id	quantity	pizza_id	pizza_type_id	size	price
1	1	hawaiian_m	1	hawaiian_m	hawaiian	M	13.25
2	2	classic_dlx_m	1	classic_dlx_m	classic_dlx	M	16
3	2	five_cheese_l	1	five_cheese_l	five_cheese	L	18.5
4	2	ital_supr_l	1	ital_supr_l	ital_supr	L	20.75
5	2	mexicana_m	1	mexicana_m	mexicana	M	16
6	2	thai_ckn_l	1	thai_ckn_l	thai_ckn	L	20.75

Self Join

A **self join** is a join where a table is joined with itself. This can be useful when you need to compare rows within the same table or create hierarchical queries. To perform a self join, you use table aliases to represent the table multiple times in the query.

```
select e1.employee_id, e1.first_name,e2.first_name as manager_name
from employees e1
join employees e2
on e1.manager_id = e2.employee_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

employee_id	first_name	manager_name
2	Jane	John
3	Alice	John
4	Bob	Jane
5	Emma	Jane

John

Calculate the total revenue generated from pizza sales.

```
SELECT
 SUM(quantity * pizzas.price) AS total_sale
FROM
 orders_details
     JOIN
 pizzas ON pizzas.pizza_id = orders_details.pizza_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sale
817860.049999993

-- for rounding up the total sale value

```
SELECT
 round(sum(quantity * pizzas.price),2) AS total_sale
FROM
 orders_details
     JOIN
 pizzas ON pizzas.pizza_id = orders_details.pizza_id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sale
817860.05

Identify the highest-priced pizza.

```
SELECT
    pizza_types.name, pizzas.price
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
ORDER BY price DESC
LIMIT 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

name	price
The Greek Pizza	35.95

Identify the most common pizza size ordered.

```sql
SELECT
    pizzas.size,
    COUNT(orders_details.order_details_id) AS total_orders
FROM
    pizzas
        JOIN
    orders_details ON pizzas.pizza_id = orders_details.pizza_id
GROUP BY pizzas.size
ORDER BY total_orders DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

size	total_orders
L	18526
M	15385
S	14137
XL	544
XXL	28

List the top 5 most ordered pizza types along with their quantities.

```sql
SELECT
    pizza_types.name,
    SUM(orders_details.quantity) AS total_order_quantity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY total_order_quantity DESC
LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

name	total_order_quantity
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Join the necessary tables to find the total quantity of each pizza category ordered.

```sql
SELECT
    pizza_types.category,
    SUM(orders_details.quantity) AS Total_order_quentity
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    orders_details ON orders_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category
ORDER BY Total_order_quentity DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

category	Total_order_quentity
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

Subquery

A subquery, also known as an inner query or nested query, is a query within another SQL query. It is used to perform operations in multiple steps by breaking down the complex query into simpler parts.

Retrieve the details of the pizza with the highest price.

```sql
SELECT pizza_id, pizza_type_id, size, price
FROM pizzas
WHERE price = (SELECT MAX(price) FROM pizzas);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

pizza_id	pizza_type_id	size	price
the_greek_xxl	the_greek	XXL	35.95