PIZZA SALES ALALYSIS

USING SQL

BY JAYENDRA NAIK

ABOUT PROJECT

The Pizza Sales Analysis project leverages SQL to explore and analyze sales data from a pizza shop. The objective is to uncover insights into sales trends, popular pizza types, and customer preferences to inform and enhance business strategies and decision-making.

TABLES

pizzas

	A	B	C	D
1	pizza_id	pizza_type_id	size	price
2	bbq_ckn_s	bbq_ckn	S	12.75
3	bbq_ckn_m	bbq_ckn	M	16.75
4	bbq_ckn_l	bbq_ckn	L	20.75
5	cali_ckn_s	cali_ckn	S	12.75
6	cali_ckn_m	cali_ckn	M	16.75
7	cali_ckn_l	cali_ckn	L	20.75
8	ckn_alfredo_s	ckn_alfredo	S	12.75
9	ckn_alfredo_m	ckn_alfredo	M	16.75
10	ckn_alfredo_l	ckn_alfredo	L	20.75

orders

	A	B	C
1	order_id	date	time
2	1	01-01-2015	11:38:36
3	2	01-01-2015	11:57:40
4	3	01-01-2015	12:12:28
5	4	01-01-2015	12:16:31
6	5	01-01-2015	12:21:30
7	6	01-01-2015	12:29:36
8	7	01-01-2015	12:50:37

order_details

	A	B	C	D
1	order_details_id	order_id	pizza_id	quantity
2	1	1	hawaiian_m	1
3	2	2	classic_dlx_m	1
4	3	2	five_cheese_l	1
5	4	2	ital_supr_l	1
6	5	2	mexicana_m	1
7	6	2	thai_ckn_l	1
8	7	3	ital_supr_m	1
9	8	3	prsc_argla_l	1
10	9	4	ital_supr_m	1
11	10	5	ital_supr_m	1
12	11	6	bbq_ckn_s	1
13	12	6	the_greek_s	1

	A	B	C	D
1	pizza_type_id	name	category	ingredients
2	bbq_ckn	The Barbecue Chicken Pizza	Chicken	Barbecued Chicken, Red Peppers, Green Peppers, Tomatoes, Red Onions, Barbecue Sauce
3	cali_ckn	The California Chicken Pizza	Chicken	Chicken, Artichoke, Spinach, Garlic, Jalapeno Peppers, Fontina Cheese, Gouda Cheese
4	ckn_alfredo	The Chicken Alfredo Pizza	Chicken	Chicken, Red Onions, Red Peppers, Mushrooms, Asiago Cheese, Alfredo Sauce
5	ckn_pesto	The Chicken Pesto Pizza	Chicken	Chicken, Tomatoes, Red Peppers, Spinach, Garlic, Pesto Sauce
6	southw_ckn	The Southwest Chicken Pizza	Chicken	Chicken, Tomatoes, Red Peppers, Red Onions, Jalapeno Peppers, Corn, Cilantro, Chipotle Sauce
7	thai_ckn	The Thai Chicken Pizza	Chicken	Chicken, Pineapple, Tomatoes, Red Peppers, Thai Sweet Chilli Sauce
8	big_meat	The Big Meat Pizza	Classic	Bacon, Pepperoni, Italian Sausage, Chorizo Sausage
9	classic_dlx	The Classic Deluxe Pizza	Classic	Pepperoni, Mushrooms, Red Onions, Red Peppers, Bacon
10	hawaiian	The Hawaiian Pizza	Classic	Sliced Ham, Pineapple, Mozzarella Cheese
11	ital_cpclio	The Italian Capocollo Pizza	Classic	Capocollo, Red Peppers, Tomatoes, Goat Cheese, Garlic, Oregano
12	napolitana	The Napolitana Pizza	Classic	Tomatoes, Anchovies, Green Olives, Red Onions, Garlic
13	pep_msh_pep	The Pepperoni, Mushroom, and Peppers Pizza	Classic	Pepperoni, Mushrooms, Green Peppers
14	pepperoni	The Pepperoni Pizza	Classic	Mozzarella Cheese, Pepperoni
15	the_greek	The Greek Pizza	Classic	Kalamata Olives, Feta Cheese, Tomatoes, Garlic, Beef Chuck Roast, Red Onions
16	brie_carre	The Brie Carre Pizza	Supreme	Brie Carre Cheese, Prosciutto, Caramelized Onions, Pears, Thyme, Garlic
17	calabrese	The Calabrese Pizza	Supreme	Nduja Salami, Pancetta, Tomatoes, Red Onions, Friggitello Peppers, Garlic
18	ital_supr	The Italian Supreme Pizza	Supreme	Calabrese Salami, Capocollo, Tomatoes, Red Onions, Green Olives, Garlic
19	peppr_salami	The Pepper Salami Pizza	Supreme	Genoa Salami, Capocollo, Pepperoni, Tomatoes, Asiago Cheese, Garlic
20	prsc_argla	The Prosciutto and Arugula Pizza	Supreme	Prosciutto di San Daniele, Arugula, Mozzarella Cheese
21	sicilian	The Sicilian Pizza	Supreme	Coarse Sicilian Salami, Tomatoes, Green Olives, Luganega Sausage, Onions, Garlic
22	soppressata	The Soppressata Pizza	Supreme	Soppressata Salami, Fontina Cheese, Mozzarella Cheese, Mushrooms, Garlic
23	spicy_ital	The Spicy Italian Pizza	Supreme	Capocollo, Tomatoes, Goat Cheese, Artichokes, Peperoncini verdi, Garlic

pizza_types

DATABASE STRUCTURE AND TABLES

```sql
create database if not exists dominos;
use dominos;
```

DATABASE CREATION


```sql
# Import all tables
select * from pizzas;
select * from pizza_types;

CREATE TABLE IF NOT EXISTS orders (
    order_id INT PRIMARY KEY,
    order_date DATE NOT NULL,
    order_time TIME NOT NULL
);
```

```sql
CREATE TABLE IF NOT EXISTS order_details (
    order_details_id INT PRIMARY KEY,
    order_id INT NOT NULL,
    pizza_id TEXT NOT NULL,
    quantity INT NOT NULL
);
```

TABLES CREATION

Tasting the small Pizza

(BASIC QUESTIONS)

BASIC QUESTIONS

1) Retrieve the total number of orders placed.

2) Calculate the total revenue generated from pizza sales.

3) Identify the highest-priced pizza.

4) Identify the most common pizza size ordered.

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

6) Identify the date with the highest number of orders placed.

7) Find the total quantity of pizzas ordered for each pizza size.

8) List the names of all unique pizza categories.

Q.1 Retrieve the total number of orders placed.

```sql
-- Q.1 Retrieve the total number of orders placed.

SELECT COUNT(order_id) AS TOTAL_ORDERS FROM orders;

```

QUERY

Result Grid | Filter Rows: | Export:

TOTAL_ORDERS
21350

OUTPUT

Q.2 Calculate the total revenue generated from pizza sales.

```sql
-- Q.2 Calculate the total revenue generated from pizza sales.

SELECT
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS TOTAL_REVENUE
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id;

```

QUERY

Result Grid | Filter Rows: | Export:

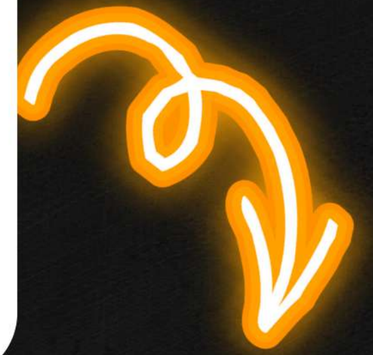
TOTAL_REVENUE
817860.05

OUTPUT

Q.3 Identify the highest-priced pizza.

```sql
-- Q.3 Identify the highest-priced pizza.

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



QUERY

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

NAME	PRICE
The Greek Pizza	35.95

OUTPUT

Q.4 Identify the most common pizza size ordered.

```sql
-- Q.4 Identify the most common pizza size ordered.

SELECT
    pizzas.size AS PIZZA_SIZE,
    COUNT(order_details.order_details_id) AS PIZZA_SIZE_ORDERED
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY pizzas.size
ORDER BY PIZZA_SIZE_ORDERED DESC
LIMIT 1;
```

QUERY

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

PIZZA_SIZE	PIZZA_SIZE_ORDERED
L	18526

OUTPUT

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

```sql
-- Q.5 List the top 5 most ordered pizza types along with their quantities.

SELECT
    pizza_types.name AS PIZZA_NAME,
    SUM(order_details.quantity) AS QUANTITIES_ORDERED
FROM
    pizza_types
        INNER JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        INNER JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.name
ORDER BY QUANTITIES_ORDERED DESC
LIMIT 5;

```

OUTPUT

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

PIZZA_NAME	QUANTITIES_ORDERED
The Classic Deluxe Pizza	2453
The Barbecue Chicken Pizza	2432
The Hawaiian Pizza	2422
The Pepperoni Pizza	2418
The Thai Chicken Pizza	2371

Q.6 Identify the date with the highest number of orders placed.

```sql
-- Q.6 Identify the date with the highest number of orders placed.

SELECT
    orders.order_date AS DATE, COUNT(orders.order_id) AS ORDERS
FROM
    orders
GROUP BY DATE
ORDER BY ORDERS DESC
LIMIT 1;

```

QUERY

OUTPUT

Result Grid | Filter Rows: | Export:

DATE	ORDERS
2015-11-27	115

Q.7 Find the total quantity of pizzas ordered for each pizza size.

```sql
-- Q.7 Find the total quantity of pizzas ordered for each pizza size.

SELECT
    pizzas.size AS SIZE, SUM(order_details.quantity) AS QUANTITY
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY SIZE
ORDER BY QUANTITY DESC;
```

QUERY

Result Grid | Filter Rows:

SIZE	QUANTITY
L	18956
M	15635
S	14403
XL	552
XXL	28

OUTPUT

Q.8 List the names of all unique pizza categories.

```sql
-- Q.8 List the names of all unique pizza categories.

SELECT DISTINCT
    pizza_types.category AS CATEGORY
FROM
    pizza_types;

```

QUERY

Result Grid | Filter Rows:

CATEGORY
Chicken
Classic
Supreme
Veggie

OUTPUT

Enjoying the Medium Pizza

(INTERMEDIATE QUESTIONS)

INTERMEDIATE QUESTIONS

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

10) Determine the distribution of orders by hour of the day.

11) Join relevant tables to find the category-wise distribution of pizzas.

12) Group the orders by date and calculate the average number of pizzas ordered per day.

13) Determine the top 3 most ordered pizza types based on revenue.

14) Determine the average revenue generated per order.

15) Calculate the total revenue generated for each pizza size.

16) Find the total number of orders placed for each day of the week.

Q.9 Join the necessary tables to find the total quantity of each pizza category ordered.

```sql
-- Q.9 Join the necessary tables to find the total quantity of each pizza category ordered.

SELECT
    pizza_types.category AS CATEGORY,
    SUM(order_details.quantity) AS QUANTITY
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY CATEGORY
ORDER BY QUANTITY DESC;

```

OUTPUT

Result Grid | Filter Rows: | Export:

CATEGORY	QUANTITY
Classic	14888
Supreme	11987
Veggie	11649
Chicken	11050

QUERY

Q.10 Determine the distribution of orders by hour of the day.

```sql
-- Q.10 Determine the distribution of orders by hour of the day.

SELECT
    HOUR(order_time) AS HOUR, COUNT(order_id) AS ORDERS
FROM
    ORDERS
GROUP BY HOUR
ORDER BY ORDERS DESC;

```

QUERY

OUTPUT

Result Grid | Filter Rows:

HOUR	ORDERS
12	2520
13	2455
18	2399
17	2336
19	2009
16	1920
20	1642
14	1472

Q.11 Join relevant tables to find the category-wise distribution of pizzas.

```sql
-- Q.11 Find the category-wise distribution of pizzas.

SELECT
    category AS CATEGORY, COUNT(name) AS COUNT_OF_NAMES
FROM
    pizza_types
GROUP BY CATEGORY
ORDER BY COUNT_OF_NAMES DESC;

```

OUTPUT

QUERY

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

CATEGORY	COUNT_OF_NAMES
Supreme	9
Veggie	9
Classic	8
Chicken	6

Q.12 Group the orders by date and calculate the average number of pizzas ordered per day.

```sql
-- Q.12 Group the orders by date and calculate the average number of pizzas ordered per day.

SELECT
    ROUND(AVG(QUANTITY),0) AS AVG_ORDER_PER_DAY
FROM
    (SELECT
        orders.order_date AS DATE,
            SUM(order_details.quantity) AS QUANTITY
    FROM
        orders
    JOIN order_details ON orders.order_id = order_details.order_id
    GROUP BY DATE) AS QUANTITY_PER_DAY;

```

QUERY

OUTPUT

Result Grid | Filter Rows: | Export:

AVG_ORDER_PER_DAY
138

Q.13 Determine the top 3 most ordered pizza types based on revenue.

```sql
-- Q.13 Determine the top 3 most ordered pizza types based on revenue.

SELECT
    pizza_types.name AS NAME,
    SUM(order_details.quantity * pizzas.price) AS REVENEU
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY NAME
ORDER BY REVENEU DESC
LIMIT 3;

```

OUTPUT

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

NAME	REVENEU
The Thai Chicken Pizza	43434.25
The Barbecue Chicken Pizza	42768
The California Chicken Pizza	41409.5

QUERY

Q.14 Determine the average revenue generated per order.

```sql
-- Q.14 Determine the average revenue generated per order.

SELECT
    ROUND(AVG(REVENUE),2) AS AVG_REVENUE_PER_ORDER
FROM
    (SELECT
        order_details.order_id,
            SUM(order_details.quantity * pizzas.price) AS REVENUE
    FROM
        order_details
    JOIN pizzas ON order_details.pizza_id = pizzas.pizza_id
    GROUP BY order_details.order_id) AS REVENUE_PER_ORDER;

```

OUTPUT

QUERY

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

AVG_REVENUE_PER_ORDER
38.31

Q.15 Calculate the total revenue generated for each pizza size.

```sql
-- Q.15 Calculate the total revenue generated for each pizza size.

SELECT
    pizzas.size AS SIZE,
    ROUND(SUM(order_details.quantity * pizzas.price),
            2) AS REVENUE
FROM
    pizzas
        JOIN
    order_details ON pizzas.pizza_id = order_details.pizza_id
GROUP BY SIZE
ORDER BY REVENUE DESC;

```

OUTPUT

Result Grid | Filter Rows:

SIZE	REVENUE
L	375318.7
M	249382.25
S	178076.5
XL	14076
XXL	1006.6

QUERY

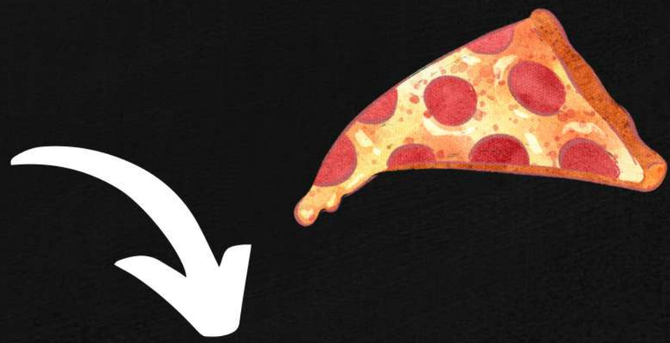
Q.16 Find the total number of orders placed for each day of the week.

```sql
-- Q.16 Find the total number of orders placed for each day of the week.

SELECT
    DAYNAME(orders.order_date) AS DAYS,
    COUNT(orders.order_id) AS NUMBER_OF_ORDERS
FROM
    orders
GROUP BY DAYS
ORDER BY FIELD(DAYS,
        'Monday',
        'Tuesday',
        'Wednesday',
        'Thursday',
        'Friday',
        'Saturday',
        'Sunday');

```

QUERY



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

DAYS	NUMBER_OF_ORDE
Monday	2794
Tuesday	2973
Wednesday	3024
Thursday	3239
Friday	3538
Saturday	3158
Sunday	2624

OUTPUT

Exploring the Large Pizza

(ADVANCED QUESTIONS)

ADVANCED QUESTIONS

17) Calculate the percentage contribution of each pizza type to total revenue.

18) Analyze the cumulative revenue generated over time.

19) Determine the top 3 most ordered pizza types based on revenue for each pizza category.

20) Calculate the monthly revenue generated from pizza sales.

21) Identify Peak Hours for Pizza Orders.

22) Identify the top 5 customers who generated the most revenue and their order details.

Q.17 Calculate the percentage contribution of each pizza type to total revenue.

QUERY

```sql
-- Q.17 Calculate the percentage contribution of each pizza type to total revenue.

SELECT
    pizza_types.category AS PIZZA_TYPES,
    CONCAT(ROUND(SUM(order_details.quantity * pizzas.price) / (SELECT
                        ROUND(SUM(order_details.quantity * pizzas.price),
                                2) AS TOTAL_REVENUE
                    FROM
                        order_details
                            JOIN
                        pizzas ON order_details.pizza_id = pizzas.pizza_id) * 100,
                2),
            ' ',
            '%') AS PERCENTAGE_OF_TOTAL_REVENUE
FROM
    pizza_types
        JOIN
    pizzas ON pizza_types.pizza_type_id = pizzas.pizza_type_id
        JOIN
    order_details ON order_details.pizza_id = pizzas.pizza_id
GROUP BY PIZZA_TYPES
ORDER BY PERCENTAGE_OF_TOTAL_REVENUE DESC;

```

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

PIZZA_TYPES	PERCENTAGE_OF_TOTAL_REVENUE
Classic	26.91 %
Supreme	25.46 %
Chicken	23.96 %
Veggie	23.68 %

OUTPUT

Q.18 Analyze the cumulative revenue generated over time.

```sql
-- Q.18 Analyze the cumulative revenue generated over time.

SELECT order_date,
SUM(REVENUE) OVER(order by order_date) AS CUMULATIVE_REVENUE
FROM
(SELECT orders.order_date,SUM(order_details.quantity * pizzas.price) AS REVENUE
FROM orders JOIN order_details
ON orders.order_id = order_details.order_id
JOIN pizzas
ON pizzas.pizza_id = order_details.pizza_id
GROUP BY orders.order_date) AS SALES;
```

QUERY

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

order_date	CUMULATIVE_REVENUE
2015-01-01	2713.8500000000004
2015-01-02	5445.75
2015-01-03	8108.15
2015-01-04	9863.6
2015-01-05	11929.55
2015-01-06	14358.5
2015-01-07	16560.7
2015-01-08	19399.05

OUTPUT

Q.19 Determine the top 3 most ordered pizza types based on revenue for each pizza category.

```sql
-- Q.19 Determine the top 3 most ordered pizza types based on revenue for each pizza category.

SELECT CATEGORY, NAME, REVENUE FROM
(SELECT CATEGORY, NAME, REVENUE,
RANK() OVER(partition by CATEGORY ORDER BY REVENUE DESC) AS RN
FROM
(SELECT pizza_types.category,pizza_types.name,
SUM(order_details.quantity * pizzas.price) AS REVENUE
FROM pizza_types JOIN pizzas
ON pizza_types.pizza_type_id = pizzas.pizza_type_id
JOIN order_details
ON order_details.pizza_id = pizzas.pizza_id
GROUP BY pizza_types.category,pizza_types.name
ORDER BY REVENUE DESC) AS A) AS B
WHERE RN <= 3;

```

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

CATEGORY	NAME	REVENUE
Chicken	The Thai Chicken Pizza	43434.25
Chicken	The Barbecue Chicken Pizza	42768
Chicken	The California Chicken Pizza	41409.5
Classic	The Classic Deluxe Pizza	38180.5
Classic	The Hawaiian Pizza	32273.25
Classic	The Pepperoni Pizza	30161.75
Supreme	The Spicy Italian Pizza	34831.25
Supreme	The Italian Supreme Pizza	33476.75
Supreme	The Sicilian Pizza	30940.5
Veggie	The Four Cheese Pizza	32265.70000000...
Veggie	The Mexicana Pizza	26780.75
Veggie	The Five Cheese Pizza	26066.5

QUERY

OUTPUT

Q.20 Calculate the monthly revenue generated from pizza sales.

```sql
-- Q.20 Calculate the monthly revenue generated from pizza sales.

SELECT
    DATE_FORMAT(orders.order_date, '%Y-%M') AS MONTHS,
    ROUND(SUM(order_details.quantity * pizzas.price),
            3) AS MONTHLY_REVENUE
FROM
    orders
        JOIN
    order_details ON orders.order_id = order_details.order_id
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id
GROUP BY MONTHS
ORDER BY MONTHS;
```

QUERY

OUTPUT

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

MONTHS	MONTHLY_REVENUE
2015-April	68736.8
2015-August	68278.25
2015-December	64701.15
2015-February	65159.6
2015-January	69793.3
2015-July	72557.9
2015-June	68230.2
2015-March	70397.1

Q.21 Identify Peak Hours for Pizza Orders.

```sql
-- Q.21 Identify Peak Hours for Pizza Orders.

SELECT
    HOUR(orders.order_time) AS HOURS_OF_A_DAY,
    SUM(order_details.quantity) AS TOTAL_PIZZAS_ORDERED
FROM
    orders
        JOIN
    order_details ON orders.order_id = order_details.order_id
GROUP BY HOURS_OF_A_DAY
ORDER BY TOTAL_PIZZAS_ORDERED DESC
LIMIT 3;

```

OUTPUT

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

HOURS_OF_A_DAY	TOTAL_PIZZAS_ORDERED
12	6776
13	6413
18	5417

QUERY

Q.22 Identify the top 5 customers who generated the most revenue and their order details.

```sql
-- Q.22 Identify the top 5 customers who generated the most revenue and their order details.

SELECT
    orders.order_id AS CUSTOMER,
    SUM(order_details.quantity * pizzas.price) AS PURCHASES,
    orders.order_date,
    orders.order_time,
    SUM(order_details.quantity) AS QUANTITY
FROM
    order_details
        JOIN
    orders ON order_details.order_id = orders.order_id
        JOIN
    pizzas ON pizzas.pizza_id = order_details.pizza_id
GROUP BY CUSTOMER
ORDER BY PURCHASES DESC
LIMIT 5;

```

QUERY

OUTPUT

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

CUSTOMER	PURCHASES	order_date	order_time	QUANTITY
18845	444.2	2015-11-18	12:25:12	28
10760	417.15	2015-06-30	13:31:27	25
1096	285.15	2015-01-19	12:56:45	15
6169	284	2015-04-14	13:14:51	15
740	280.95	2015-01-13	12:29:51	15

jayendranaik00@gmail.com

https://github.com/jayendranaik

www.linkedin.com/in/jayendranaik032002