Experiment-1

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv("pizza_sales.csv")
df.head(3)
```

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category	pizza_ingredients	pizza_name
0	1.0	1.0	hawaiian_m	1.0	1/1/2015	11:38:36	13.25	13.25	M	Classic	Sliced Ham, Pineapple, Mozzarella Cheese	The Hawaiian Pizza
1	2.0	2.0	classic_dlx_m	1.0	1/1/2015	11:57:40	16.00	16.00	M	Classic	Pepperoni, Mushrooms, Red Onions, Red Pepper...	The Classic Deluxe Pizza
2	3.0	2.0	five_cheese_l	1.0	1/1/2015	11:57:40	18.50	18.50	L	Veggie	Mozzarella Cheese, Provolone Cheese, Smoked Go...	The Five Cheese Pizza

```
df.isnull().head()
```

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category	pizza_ingredients	pizza_name
0	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False

```
df['pizza_id'].isnull().sum()
```

```
0
```

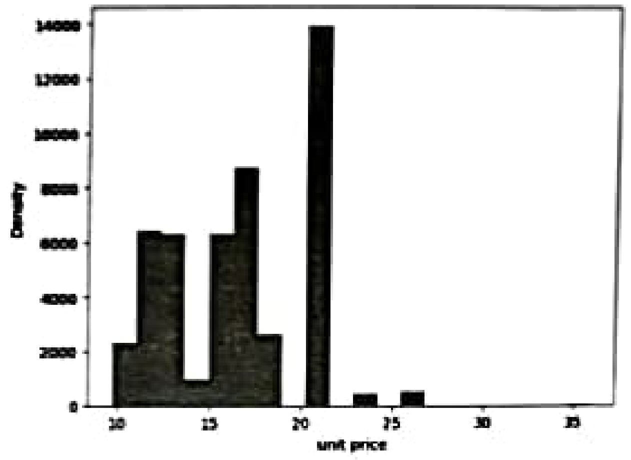
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48620 entries, 0 to 48619
Data columns (total 12 columns):
 #  Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0  pizza_id         48620 non-null  float64
 1  order_id         48620 non-null  float64
 2  pizza_name_id    48620 non-null  object
 3  quantity         48620 non-null  float64
 4  order_date       48620 non-null  object
 5  order_time       48620 non-null  object
 6  unit_price       48620 non-null  float64
 7  total_price      48620 non-null  float64
 8  pizza_size       48620 non-null  object
 9  pizza_category   48620 non-null  object
 10 pizza_ingredients 48620 non-null  object
 11 pizza_name       48620 non-null  object
dtypes: float64(5), object(7)
```

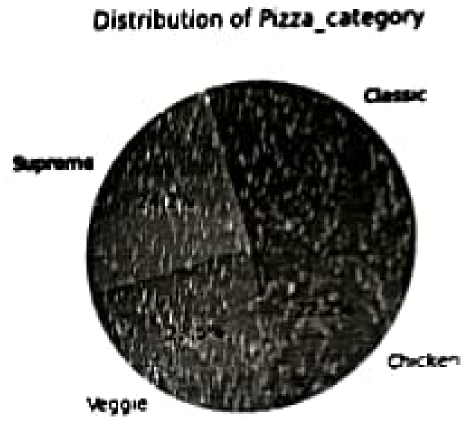
```
df.describe()
```

	pizza_id	order_id	quantity	unit_price	total_price
count	48620.000000	48620.000000	48620.000000	48620.000000	48620.000000
mean	24310.500000	10701.479761	1.019622	16.494132	16.821474
std	14035.529381	6180.119770	0.143077	3.621789	4.437398
min	1.000000	1.000000	1.000000	9.750000	9.750000
25%	12155.750000	5337.000000	1.000000	12.750000	12.750000
50%	24310.500000	10682.500000	1.000000	16.500000	16.500000
75%	36465.250000	16100.000000	1.000000	20.250000	20.500000
max	48620.000000	21350.000000	4.000000	35.950000	83.000000

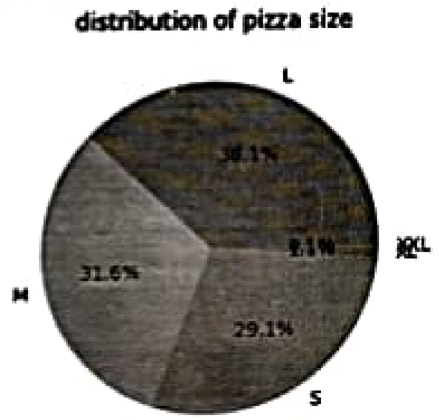
```
plt.hist(df['unit_price'],bins=20)
plt.xlabel('unit price')
plt.ylabel('Density')
plt.show()
```



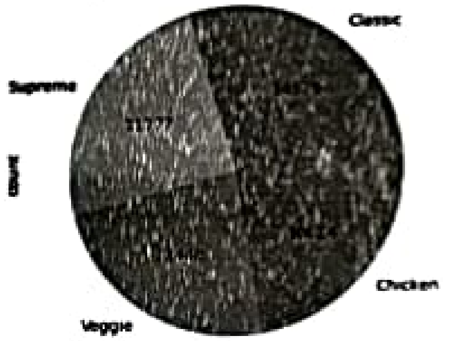
```
counts=df['pizza_category'].value_counts()
plt.figure(figsize=(4,4))
plt.pie(counts,labels=counts.index,
autopct='%1.1f%%')
plt.title("Distribution of Pizza_category")
plt.show()
```



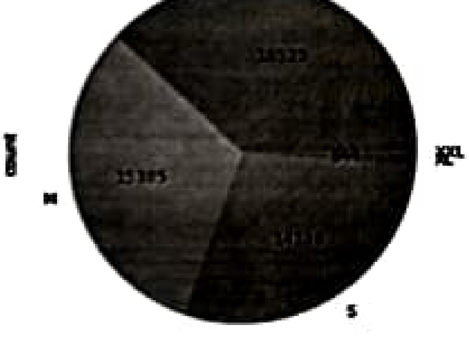
```
counts=df['pizza_size'].value_counts()
plt.figure(figsize=(4,4))
plt.pie(counts,labels=counts.index, autopct='%1.1f%%')
plt.title("distribution of pizza size")
plt.show()
```



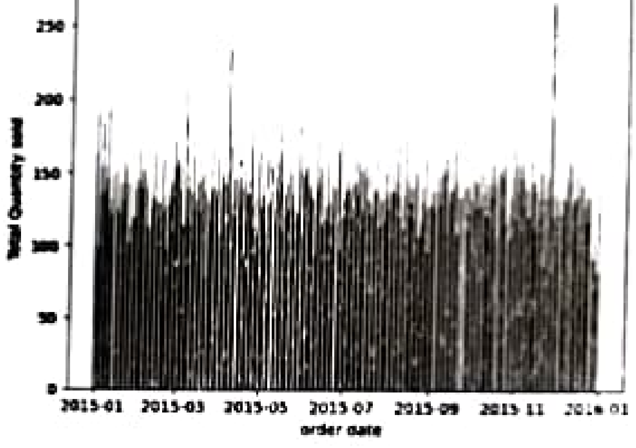
```
counts=df['pizza_category'].value_counts()
counts.plot.pie(autopct=lambda p:
f'{int(p*sum(counts)/100)}')
plt.show()
```



```
counts=df['pizza_size'].value_counts()
counts.plot.pie(autopct=lambda p:
f'{int(p*sum(counts)/100)}')
plt.show()
```



```
df['order_date'] = pd.to_datetime(df['order_date'],
format='mixed')
sales_by_date=df.groupby('order_date')['quantity'].sum()
plt.bar(sales_by_date.index,sales_by_date.values)
plt.xlabel('order date')
plt.ylabel('Total Quantity sold')
plt.show()
```



/

ASSIGNMENT 3

```
CREATE TABLE locations(
  city_id int primary key,
  city varchar(20),
  state varchar(20),
  country varchar(20)
);
INSERT INTO locations (city_id, city, state, country)
VALUES
(101, 'Kolkata', 'West Bengal', 'India');
INSERT INTO locations (city_id, city, state, country)
VALUES
(102, 'Ahmadabad', 'Gujrat', 'India');
INSERT INTO locations (city_id, city, state, country)
VALUES
(103, 'Barrackpore', 'West Bengal', 'India');
INSERT INTO locations (city_id, city, state, country)
VALUES
(104, 'Gandhinagar', 'Gujrat', 'India');
CREATE TABLE DateTable(
  Date_id int primary key,
  Day int ,
  month varchar(20),
  year int
);
INSERT INTO DateTable(Date_id,Day,Month,year)
VALUES(105,12,'February',2025);
INSERT INTO DateTable(Date_id,Day,Month,year)
VALUES(106,13,'March',2023);
INSERT INTO DateTable(Date_id,Day,Month,year)
VALUES(107,15,'April',2025);
INSERT INTO DateTable(Date_id,Day,Month,year)
VALUES(108,16,'May',2023);
CREATE TABLE Products(
  Product_id int primary key,
  Product_name varchar(20),
  Product_tag varchar(20),
  Product_price int
);
INSERT INTO Products(Product_id,Product_name,Product_tag,Product_price)
VALUES(201,'Chalk','58D192',100);
INSERT INTO Products(Product_id,Product_name,Product_tag,Product_price)
VALUES(202,'Cosmetics','1P46V93',250);
INSERT INTO Products(Product_id,Product_name,Product_tag,Product_price)
VALUES(203,'Computer','P86V92',1500);
INSERT INTO Products(Product_id,Product_name,Product_tag,Product_price)
VALUES(204,'Watch','K36M92',300);
CREATE TABLE new_sales(
   Sales_id INT PRIMARY KEY,
   city_id INT,
   Date_id INT,
   Product_id INT,
   Total_Sales_Amount INT,
   FOREIGN KEY (city_id) REFERENCES locations(city_id),
   FOREIGN KEY (Date_id) REFERENCES DateTable(Date_id),
   FOREIGN KEY (Product_id) REFERENCES Products(Product_id)
);
CREATE OR REPLACE TRIGGER sales_id_trigger
BEFORE INSERT ON new_sales
FOR EACH ROW
DECLARE
```

```
 v_sales_id INT;
BEGIN
 -- Get the next value from the sequence
 SELECT sales_id_seq.NEXTVAL
 INTO v_sales_id
 FROM dual;

 -- Assign the value to :new.Sales_id
 :new.Sales_Id := v_sales_id;
END;
INSERT INTO new_sales (city_id, Date_id, Product_id, Total_Sales_Amount)
VALUES (101, 105, 201, 100);
INSERT INTO new_sales (city_id, Date_id, Product_id, Total_Sales_Amount)
VALUES (102, 106, 202, 250);
INSERT INTO new_sales (city_id, Date_id, Product_id, Total_Sales_Amount)
VALUES (103, 107, 203, 1500);
INSERT INTO new_sales (city_id, Date_id, Product_id, Total_Sales_Amount)
VALUES (104, 108, 204, 300);
BEGIN
  FOR i IN 1..64 LOOP
    INSERT INTO new_sales (city_id, Date_id, Product_id, Total_Sales_Amount)
    VALUES (
      MOD(i, 4) + 101,  -- Cycles through city_id 101 to 104
      MOD(i, 4) + 105,  -- Cycles through Date_id 105 to 108
      MOD(i, 4) + 201,  -- Cycles through Product_id 201 to 204
      100 * (MOD(i, 4) + 1)  -- Increases Total_Sales_Amount (100, 200, 300, 400)
    );
  END LOOP;
  COMMIT;
END;
select * from new_sales;
```

OUTPUT TABLE

SALES_ID	CITY_ID	DATE_ID	PRODUCT_ID	TOTAL_SALES
1	101	105	201	100
2	102	106	202	250
3	102	106	202	250
4	103	107	203	1500
5	104	108	204	300
6	102	106	202	200
7	103	107	203	300
8	104	108	204	400
9	101	105	201	100
10	102	106	202	200
11	103	107	203	300
12	104	108	204	400
13	101	105	201	100
14	102	106	202	200
15	103	107	203	300
16	104	108	204	400
17	101	105	201	100
18	102	106	202	200
19	103	107	203	300
20	104	108	204	400
21	101	105	201	100
22	102	106	202	200

23	103	107	203	300
24	104	108	204	400
25	101	105	201	100
26	102	106	202	200
27	103	107	203	300
28	104	108	204	400
29	101	105	201	100
30	102	106	202	200
31	103	107	203	300
32	104	108	204	400
33	101	105	201	100
34	102	106	202	200
35	103	107	203	300
36	104	108	204	400
37	101	105	201	100
38	102	106	202	200
39	103	107	203	300
40	104	108	204	400
41	101	105	201	100
42	102	106	202	200
43	103	107	203	300
44	104	108	204	400
45	101	105	201	100
46	102	106	202	200
47	103	107	203	300
48	104	108	204	400
49	101	105	201	100
50	102	106	202	200
51	103	107	203	300
52	104	108	204	400
53	101	105	201	100
54	102	106	202	200
55	103	107	203	300
56	104	108	204	400
57	101	105	201	100
58	102	106	202	200
59	103	107	203	300
60	104	108	204	400
61	101	105	201	100
62	102	106	202	200
63	103	107	203	300

/

ASSIGNMENT 4

```
CREATE TABLE PRODUCT_TABLE (
   PRODUCT_ID NUMBER PRIMARY KEY,
   PRODUCT_NAME VARCHAR2(50),
   CATEGORY VARCHAR2(30),
   PRICE NUMBER(10,2)
);
CREATE TABLE CUSTOMER_TABLE (
   CUSTOMER_ID NUMBER PRIMARY KEY,
   CUSTOMER_NAME VARCHAR2(50),
   CITY VARCHAR2(30),
   EMAIL VARCHAR2(50)
);
CREATE TABLE SALES_TABLE (
   SALES_ID NUMBER PRIMARY KEY,
   PRODUCT_ID NUMBER,
   CUSTOMER_ID NUMBER,
   QUANTITY NUMBER,
   SALE_DATE DATE,
   FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT_TABLE(PRODUCT_ID),
   FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER_TABLE(CUSTOMER_ID)
);
INSERT INTO PRODUCT_TABLE VALUES (1, 'Laptop', 'Electronics', 1200.00);
INSERT INTO PRODUCT_TABLE VALUES (2, 'Smartphone', 'Electronics', 800.00);
INSERT INTO PRODUCT_TABLE VALUES (3, 'Tablet', 'Electronics', 500.00);
INSERT INTO PRODUCT_TABLE VALUES (4, 'Smartwatch', 'Wearables', 200.00);

INSERT INTO CUSTOMER_TABLE VALUES (101, 'John Doe', 'New York', 'john@example.com');
INSERT INTO CUSTOMER_TABLE VALUES (102, 'Jane Smith', 'Los Angeles', 'jane@example.com');
INSERT INTO CUSTOMER_TABLE VALUES (103, 'Robert Brown', 'Chicago', 'robert@example.com');
INSERT INTO CUSTOMER_TABLE VALUES (104, 'Emily Johnson', 'Houston', 'emily@example.com');

INSERT INTO SALES_TABLE VALUES (1001, 1, 101, 2, TO_DATE('2024-02-10', 'YYYY-MM-DD'));
INSERT INTO SALES_TABLE VALUES (1002, 2, 102, 1, TO_DATE('2024-02-12', 'YYYY-MM-DD'));
INSERT INTO SALES_TABLE VALUES (1003, 3, 103, 3, TO_DATE('2024-02-15', 'YYYY-MM-DD'));
INSERT INTO SALES_TABLE VALUES (1004, 4, 104, 1, TO_DATE('2024-02-18', 'YYYY-MM-DD'));
Verifying tables
SELECT * FROM PRODUCT_TABLE;
SELECT * FROM CUSTOMER_TABLE;
SELECT * FROM SALES_TABLE;
```

-- Roll UP

```
select s.PRODUCT_ID, sum(s.quantity) as total_quantity,
 sum(s.quantity*p.price) as total_sales
from sales_table s
join product_table p on s.product_id=p.product_id
group by rollup(s.product_id);
```

PRODUCT_ID	TOTAL_QUANTITY	TOTAL_SALES
1	2	2400
2	1	800
3	3	1500
4	1	200
	7	4900

--DRILL DOWN

```
select
s.product_id,c.customer_id,c.customer_name,sum
(s.QUANTITY) as total_quantity
from sales_table s join customer_table c on
s.customer_id=c.customer_id
group by
s.PRODUCT_ID,c.customer_id,c.customer_name
order by s.PRODUCT_ID,c.customer_id;
```

PRODUCT_ID	CUSTOMER_ID	CUSTOMER_NAME	TOTAL_QUANTITY
1	101	John Doe	2
2	102	Jane Smith	1
3	103	Robert Brown	3
4	104	Emily Johnson	1

--SLICE

Retrieves sales data only for Product_ID=1(Laptop)

```
SELECT * FROM SALES_TABEL WHERE PRODUCT_ID=1;
```

SALES_ID	PRODUCT_ID	CUSTOMER_ID	QUANTITY	SALE_DATE
1001	1	101	2	2024-02-10

--Dice

Retrieves sales data for:
PRODUCT)ID=1(LAPTOP)
CUSTOMER_ID=101 (JOHN DOE)

```
SELECT * FROM SALES_TABLE WHERE PRODUCT_ID=1 AND CUSTOMER_ID=101;
```

SALES_ID	PRODUCT_ID	CUSTOMER_ID	QUANTITY	SALE_DATE
1001	1	101	2	2024-02-10