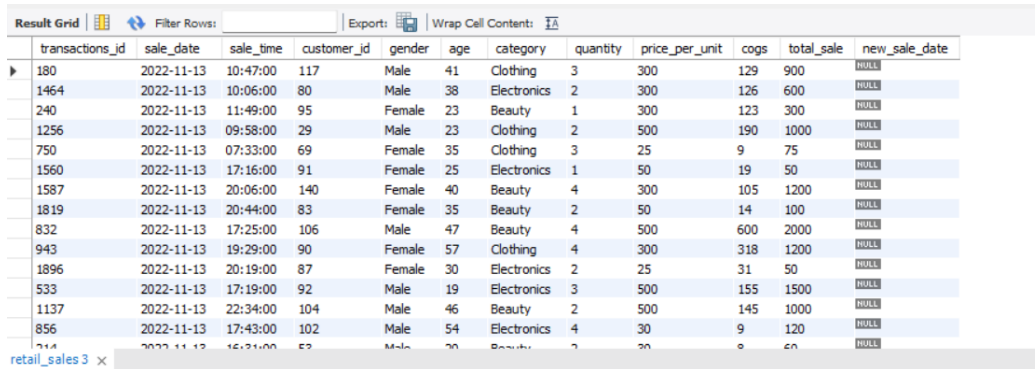


Data Analysis and Findings

1. Write a SQL query to retrieve all columns for sales made on '2022-11-13'.

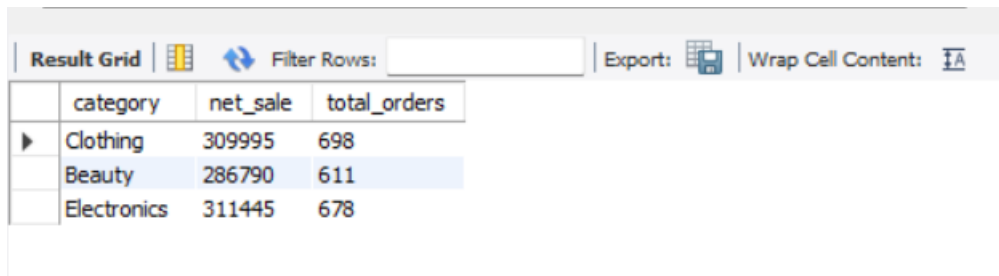
```
SELECT *  
FROM retail_sales  
WHERE sale_date = '2022-11-13';
```



transactions_id	sale_date	sale_time	customer_id	gender	age	category	quantity	price_per_unit	cogs	total_sale	new_sale_date
180	2022-11-13	10:47:00	117	Male	41	Clothing	3	300	129	900	NULL
1464	2022-11-13	10:06:00	80	Male	38	Electronics	2	300	126	600	NULL
240	2022-11-13	11:49:00	95	Female	23	Beauty	1	300	123	300	NULL
1256	2022-11-13	09:58:00	29	Male	23	Clothing	2	500	190	1000	NULL
750	2022-11-13	07:33:00	69	Female	35	Clothing	3	25	9	75	NULL
1560	2022-11-13	17:16:00	91	Female	25	Electronics	1	50	19	50	NULL
1587	2022-11-13	20:06:00	140	Female	40	Beauty	4	300	105	1200	NULL
1819	2022-11-13	20:44:00	83	Female	35	Beauty	2	50	14	100	NULL
832	2022-11-13	17:25:00	106	Male	47	Beauty	4	500	600	2000	NULL
943	2022-11-13	19:29:00	90	Female	57	Clothing	4	300	318	1200	NULL
1896	2022-11-13	20:19:00	87	Female	30	Electronics	2	25	31	50	NULL
533	2022-11-13	17:19:00	92	Male	19	Electronics	3	500	155	1500	NULL
1137	2022-11-13	22:34:00	104	Male	46	Beauty	2	500	145	1000	NULL
856	2022-11-13	17:43:00	102	Male	54	Electronics	4	30	9	120	NULL
714	2022-11-13	16:21:00	52	Male	20	Beauty	2	20	8	60	NULL

2. Write a SQL query to calculate the total sales (total_sale) for each category.

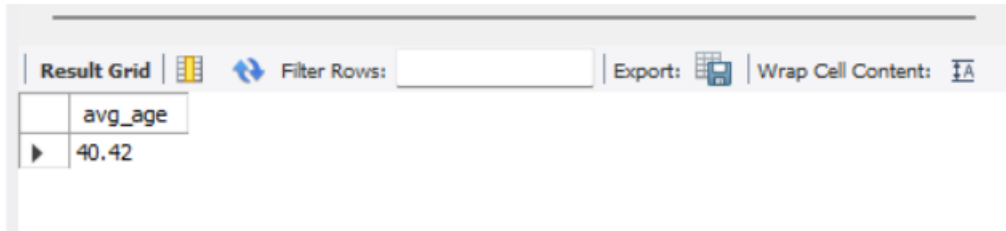
```
SELECT  
    category,  
    SUM(total_sale) as net_sale,  
    COUNT(*) as total_orders  
FROM retail_sales  
GROUP BY 1
```



category	net_sale	total_orders
Clothing	309995	698
Beauty	286790	611
Electronics	311445	678

3. Write a SQL query to find the average age of customers who purchased items from the 'Beauty' category.

```
SELECT  
    ROUND(AVG(age), 2) as avg_age  
FROM retail_sales  
WHERE category = 'Beauty'
```

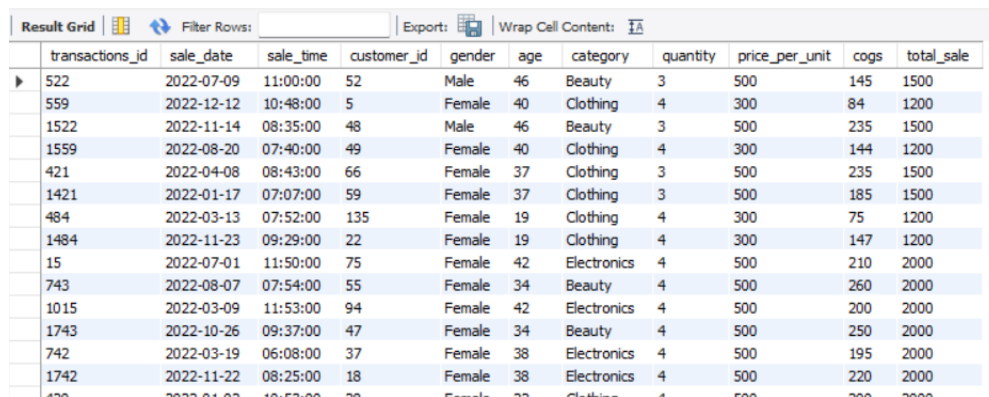


The screenshot shows a database interface with a toolbar at the top containing 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, a small table displays the result of the query:

	avg_age
▶	40.42

4. Write a SQL query to find all transactions where the total_sale is greater than 1000.

```
SELECT * FROM retail_sales  
WHERE total_sale > 1000;
```



The screenshot shows a database interface with a toolbar at the top containing 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. Below the toolbar, a table displays the results of the query:

	transactions_id	sale_date	sale_time	customer_id	gender	age	category	quantity	price_per_unit	cogs	total_sale
▶	522	2022-07-09	11:00:00	52	Male	46	Beauty	3	500	145	1500
	559	2022-12-12	10:48:00	5	Female	40	Clothing	4	300	84	1200
	1522	2022-11-14	08:35:00	48	Male	46	Beauty	3	500	235	1500
	1559	2022-08-20	07:40:00	49	Female	40	Clothing	4	300	144	1200
	421	2022-04-08	08:43:00	66	Female	37	Clothing	3	500	235	1500
	1421	2022-01-17	07:07:00	59	Female	37	Clothing	3	500	185	1500
	484	2022-03-13	07:52:00	135	Female	19	Clothing	4	300	75	1200
	1484	2022-11-23	09:29:00	22	Female	19	Clothing	4	300	147	1200
	15	2022-07-01	11:50:00	75	Female	42	Electronics	4	500	210	2000
	743	2022-08-07	07:54:00	55	Female	34	Beauty	4	500	260	2000
	1015	2022-03-09	11:53:00	94	Female	42	Electronics	4	500	200	2000
	1743	2022-10-26	09:37:00	47	Female	34	Beauty	4	500	250	2000
	742	2022-03-19	06:08:00	37	Female	38	Electronics	4	500	195	2000
	1742	2022-11-22	08:25:00	18	Female	38	Electronics	4	500	220	2000
	420	2022-01-07	10:53:00	78	Female	22	Clothing	4	500	200	2000

5. Write a SQL query to find the total number of transactions (transaction_id) made by each gender in each category.

```
select category , gender , Count(*) from retail_sales
```

```
group by gender , category ;
```

Result Grid			Filter Rows: <input type="text"/>	Export:	Wrap Cell Content:
	category	gender	Count(*)		
▶	Clothing	Male	351		
	Beauty	Male	281		
	Clothing	Female	347		
	Electronics	Male	343		
	Beauty	Female	330		
	Electronics	Female	335		

6. Write a SQL query to calculate the average sale for each month. Find out best selling month in each year.

```
SELECT
```

```
EXTRACT(YEAR FROM sale_date) AS year,
```

```
EXTRACT(MONTH FROM sale_date) AS month,
```

```
AVG(total_sale) AS avg_sal,
```

```
RANK() OVER (
```

```
PARTITION BY EXTRACT(YEAR FROM sale_date)
```

```
ORDER BY AVG(total_sale) DESC
```

```
) AS rank_
```

```
FROM retail_sales
```

```
GROUP BY EXTRACT(YEAR FROM sale_date), EXTRACT(MONTH FROM sale_date)
```

```
order by avg_sal desc;
```

#second method

```
WITH monthly_avg_sales AS (
```

```
SELECT
```

```
EXTRACT(YEAR FROM sale_date) AS year,
```

```
EXTRACT(MONTH FROM sale_date) AS month,
```

```
AVG(total_sale) AS avg_monthly_sale
```

```

FROM retail_sales

GROUP BY EXTRACT(YEAR FROM sale_date), EXTRACT(MONTH FROM sale_date)

),

ranked_months AS (

SELECT *,

    RANK() OVER (

        PARTITION BY year

        ORDER BY avg_monthly_sale DESC

    ) AS rank_

FROM monthly_avg_sales

)

SELECT year, month, avg_monthly_sale

FROM ranked_months

WHERE rank_ = 1;

```

Result Grid					Filter Rows:		Export:	Wrap Cell Content:
	year	month	avg_sal	rank_				
▶	2022	7	541.3415	1				
	2023	2	535.5319	1				
	2022	3	521.2222	2				
	2022	4	500.6140	3				
	2023	8	495.9649	2				
	2023	12	490.3901	3				
	2022	9	485.1969	4				
	2022	6	481.3953	5				
	2022	5	480.0000	6				
	2022	11	472.0205	7				
	2022	10	467.1379	8				
	2023	4	466.4894	4				
	2023	9	462.7397	5				
	2022	12	460.7692	9				

7. Write a SQL query to find the top 5 customers based on the highest total sales.

```
SELECT
    customer_id,
    SUM(total_sale) as total_sales
FROM retail_sales
GROUP BY 1
ORDER BY 2 DESC
LIMIT 5;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:	Fetch rows:
customer_id	total_sales			
3	38440			
1	30750			
5	30405			
2	25295			
4	23580			

8. Write a SQL query to find the number of unique customers who purchased items from each category.

```
SELECT
    category,
    COUNT(DISTINCT customer_id) as cnt_unique_cs
FROM retail_sales
GROUP BY category;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
category	cnt_unique_cs		
Beauty	141		
Clothing	149		
Electronics	144		

9. Write a SQL query to create each shift and number of orders.

```
with hourly_sale as (  
  select * ,  
  case  
    when extract(hour from sale_time) < 12 then "Morning"  
    when extract(hour from sale_time) between 12 and 17 then "afternoon"  
    else "evening"  
  end as shift  
  from retail_sales  
)  
select shift , count(*) as total_orders  
from hourly_sale  
group by shift;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	shift	total_orders		
▶	Morning	548		
	evening	1062		
	afternoon	377		

10. Monthly Sales Growth/Decline



Calculate the month-over-month percentage change in total_sale for each category.

```
with Monthly_sales as (  
  select  
    extract(year from sale_date) as year ,  
    extract(month from sale_date) as month ,  
    category , Sum(total_sale) as total_sales  
  from retail_sales  
  group by extract(year from sale_date) ,  
    extract(month from sale_date) ,  
    category  
  
  ) ,  
lag_month as (  
  select year , month , category ,  
    total_sales ,  
    lag(total_sales ,1,0) over (partition by category order by year , month ) as previous_sales  
  from Monthly_sales  
)
```

```

SELECT
    year ,
    month,
    category,
    previous_sales ,
    case
        when previous_sales > 0 then ((total_sales - previous_sales)/ previous_sales) * 100
        else null
    end as percentage_chng
from
    Lag_month;

```

Result Grid					
Filter Rows:		Export:  Wrap Cell Content: 			
	year	month	category	previous_sales	percentage_chng
▶	2022	1	Beauty	0	NULL
	2022	2	Beauty	8015	-28.0100
	2022	3	Beauty	5770	66.8111
	2022	4	Beauty	9625	13.2468
	2022	5	Beauty	10900	5.1376
	2022	6	Beauty	11460	-43.6300
	2022	7	Beauty	6460	10.7585
	2022	8	Beauty	7155	-2.0964
	2022	9	Beauty	7005	132.3340
	2022	10	Beauty	16275	50.7834
	2022	11	Beauty	24540	-19.4988
	2022	12	Beauty	19755	24.0192
	2023	1	Beauty	24500	-80.0408
	2023	2	Beauty	4890	112.7812

11. Gender-Based Purchasing Habits

Identify the top 3 categories by total_sale for each gender.

```

select
    gender , category ,
    sum(total_sale) as total_sales ,
    rank() over(partition by gender order by sum(total_sale) desc) as rank_
from retail_sales

```

group by gender ,category ;

Result Grid					Filter Rows:		Export:	Wrap Cell Content:
	gender	category	total_sales	rank_				
▶	Female	Clothing	162460	1				
	Female	Electronics	151180	2				
	Female	Beauty	149470	3				
	Male	Electronics	160265	1				
	Male	Clothing	147535	2				
	Male	Beauty	137320	3				

12. .Sales in peak hour.

```
select extract(hour from sale_time) as sales_hour ,  
sum(total_sale) as total_sales ,  
rank () over(order by sum(total_sale) desc ) as rnk  
from retail_sales  
group by sales_hour  
order by total_sales desc ;
```

Result Grid				Filter Rows:		Export:	Wrap Cell Content:
	sales_hour	total_sales	rnk				
▶	19	109460	1				
	21	97650	2				
	17	96480	3				
	20	93035	4				
	18	91090	5				
	22	82245	6				
	8	46430	7				
	7	46125	8				
	11	45875	9				
	10	42730	10				
	9	38255	11				
	6	36995	12				
	14	18900	13				
	16	16965	14				