

SQL for Data Analysis

Objective: Use SQL queries to extract and analyze data from a database.

Tools: Oracle (Used Oracle as it was available on my laptop).

Deliverables: SQL queries in a SQL file + screenshots of output

Dataset: This SQL task was performed using Oracle Database.

- Database Name: Raj_task_3
- Tables Used: products, orders, order_items, customers, categories, payments

Task Summary:

In this task, an eCommerce database was created using Oracle SQL with tables like **customers**, **products**, **orders**, and **payments**. Realistic data was inserted, and various SQL queries were executed using **SELECT**, **JOIN**, **GROUP BY**, **SUM**, and **AVG** to analyze sales, customer behavior, and product performance.

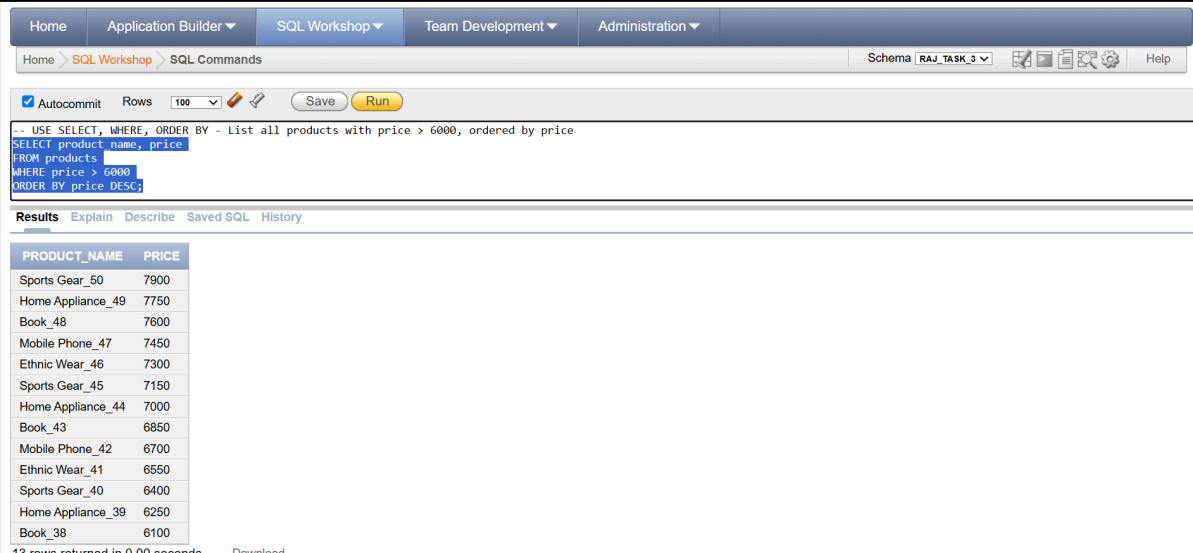
----- Perform Task and Screenshots -----

-- 1. USE SELECT, WHERE, ORDER BY List all products with price 6000, ordered by price

```
SELECT
    product_name,
    price
FROM
    products
WHERE
```

```
price > 6000  
ORDER BY  
price DESC;
```

Output:



The screenshot shows the SQL Workshop interface with a query editor and a results pane. The query editor contains the following SQL code:

```
-- USE SELECT, WHERE, ORDER BY - List all products with price > 6000, ordered by price  
SELECT product_name, price  
FROM products  
WHERE price > 6000  
ORDER BY price DESC;
```

The results pane displays the following table:

PRODUCT_NAME	PRICE
Sports Gear_50	7900
Home Appliance_49	7750
Book_48	7600
Mobile Phone_47	7450
Ethnic Wear_46	7300
Sports Gear_45	7150
Home Appliance_44	7000
Book_43	6850
Mobile Phone_42	6700
Ethnic Wear_41	6550
Sports Gear_40	6400
Home Appliance_39	6250
Book_38	6100

13 rows returned in 0.00 seconds

-- 2. GROUP BY with Aggregate Functions - Total sales by category

```
SELECT  
    c.category_name,  
    SUM(p.price * oi.quantity) AS total_sales  
FROM  
    order_items oi  
JOIN  
    products p ON oi.product_id = p.product_id  
JOIN  
    categories c ON p.category_id = c.category_id  
GROUP BY  
    c.category_name  
ORDER BY  
    total_sales DESC;
```

Output:

Oracle Application Express interface showing a SQL query in the SQL Commands window. The query calculates total sales by category using a GROUP BY clause. The results are displayed in a table below the query editor.

```
-- GROUP BY with Aggregate Functions - Total sales by category
SELECT c.category_name, SUM(p.price * oi.quantity) AS total_sales
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
JOIN categories c ON p.category_id = c.category_id
GROUP BY c.category_name
ORDER BY total_sales DESC;
```

CATEGORY_NAME	TOTAL_SALES
Stationery	149000
Sports	122750
Beauty Products	115000
Home Appliances	94000
Toys	83100
Books	67350
Furniture	53300
Clothing	42800
Groceries	25800
Electronics	20350

10 rows returned in 0.00 seconds

/*----- Use JOINS (INNER, LEFT, RIGHT) -----*/

-- 3.1 INNER JOIN: Orders with customer info

```
SELECT o.order_id, o.order_date, c.customer_name
```

```
FROM orders o
```

```
JOIN customers c ON o.customer_id = c.customer_id;
```

Output:

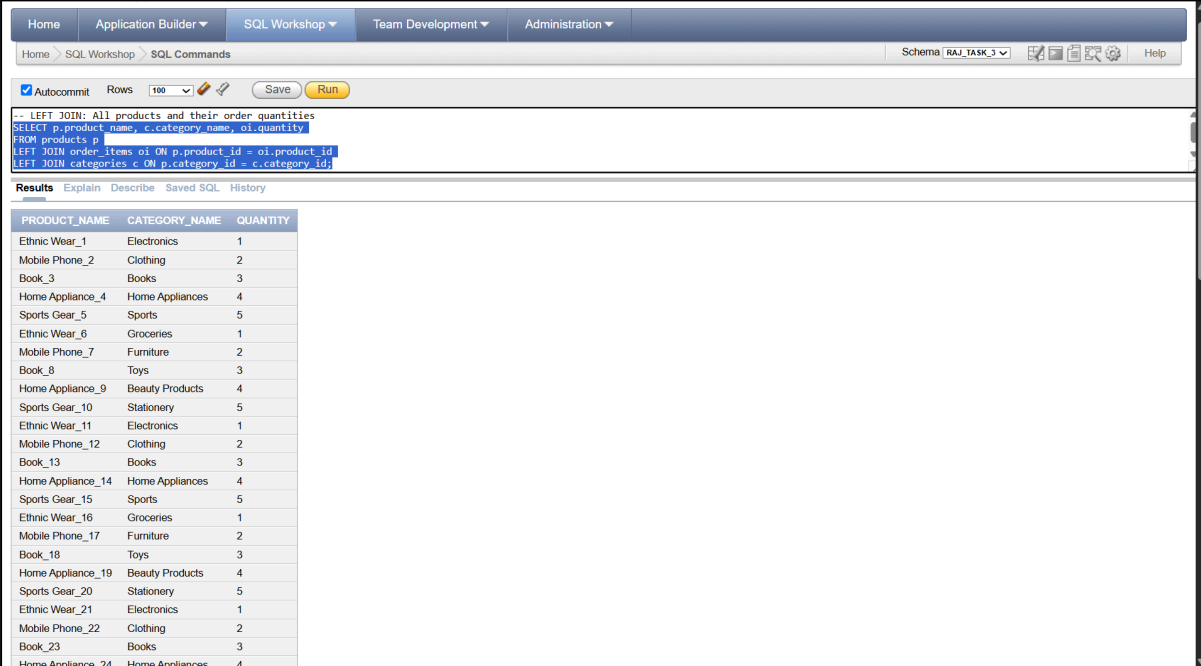
Oracle Application Express interface showing an INNER JOIN query in the SQL Commands window. The query retrieves order details along with customer names. The results are displayed in a table below the query editor.

```
-- INNER JOIN: Orders with customer info
SELECT o.order_id, o.order_date, c.customer_name
FROM orders o
JOIN customers c ON o.customer_id = c.customer_id;
```

ORDER_ID	ORDER_DATE	CUSTOMER_NAME
101	01/02/2025	Raj Jaiswal
102	01/03/2025	Ananya Gupta
103	01/04/2025	Rahul Singh
104	01/05/2025	Priya Patel
105	01/06/2025	Arjun Kumar
106	01/07/2025	Kavya Mehta
107	01/08/2025	Rohan Joshi
108	01/09/2025	Sneha Nair
109	01/10/2025	Vivek Reddy
110	01/11/2025	Divya Iyer
111	01/12/2025	Ishaan Malhotra
112	01/13/2025	Anjali Desai
113	01/14/2025	Manish Verma
114	01/15/2025	Neha Rao
115	01/16/2025	Karan Kapoor
116	01/17/2025	Pooja Joshi
117	01/18/2025	Siddharth Bhat
118	01/19/2025	Riya Sharma
119	01/20/2025	Aditya Choudhary
120	01/21/2025	Meera Singh
121	01/22/2025	Harsh Patel
122	01/23/2025	Nisha Iyer
123	01/24/2025	Sameer Nair
124	01/25/2025	Anushka Reddy

```
-- 3.2 LEFT JOIN: All products and their order quantities
SELECT p.product_name, c.category_name, oi.quantity
FROM products p
LEFT JOIN order_items oi ON p.product_id = oi.product_id
LEFT JOIN categories c ON p.category_id = c.category_id;
```

Output:



The screenshot shows the SQL Workshop interface with the following query executed:

```
-- LEFT JOIN: All products and their order quantities
SELECT p.product_name, c.category_name, oi.quantity
FROM products p
LEFT JOIN order_items oi ON p.product_id = oi.product_id
LEFT JOIN categories c ON p.category_id = c.category_id;
```

The results are displayed in a table with the following columns: PRODUCT_NAME, CATEGORY_NAME, and QUANTITY. The table contains 24 rows of data.

PRODUCT_NAME	CATEGORY_NAME	QUANTITY
Ethnic Wear_1	Electronics	1
Mobile Phone_2	Clothing	2
Book_3	Books	3
Home Appliance_4	Home Appliances	4
Sports Gear_5	Sports	5
Ethnic Wear_6	Groceries	1
Mobile Phone_7	Furniture	2
Book_8	Toys	3
Home Appliance_9	Beauty Products	4
Sports Gear_10	Stationery	5
Ethnic Wear_11	Electronics	1
Mobile Phone_12	Clothing	2
Book_13	Books	3
Home Appliance_14	Home Appliances	4
Sports Gear_15	Sports	5
Ethnic Wear_16	Groceries	1
Mobile Phone_17	Furniture	2
Book_18	Toys	3
Home Appliance_19	Beauty Products	4
Sports Gear_20	Stationery	5
Ethnic Wear_21	Electronics	1
Mobile Phone_22	Clothing	2
Book_23	Books	3
Home Appliance_24	Home Appliances	4

```
-- 3.3 RIGHT JOIN using LEFT JOIN (by swapping tables):
All order items and include product info
SELECT oi.order_id, oi.product_id, p.product_name,
p.price
FROM products p
RIGHT JOIN order_items oi ON p.product_id =
oi.product_id;
```

Output:

Home Application Builder SQL Workshop Team Development Administration				
Home > SQL Workshop > SQL Commands				
Schema: RAJ_TASK_3				
Autocommit Rows: 100 Save Run				
<pre>-- RIGHT JOIN using LEFT JOIN (by swapping tables): All order items and include product info SELECT oi.order_id, oi.product_id, p.product_name, p.price FROM products p RIGHT JOIN order_items oi ON p.product_id = oi.product_id;</pre>				
Results Explain Describe Saved SQL History				
ORDER_ID	PRODUCT_ID	PRODUCT_NAME	PRICE	
106	1	Ethnic Wear_1	550	
101	1	Ethnic Wear_1	550	
107	2	Mobile Phone_2	700	
102	2	Mobile Phone_2	700	
108	3	Book_3	850	
103	3	Book_3	850	
109	4	Home Appliance_4	1000	
104	4	Home Appliance_4	1000	
110	5	Sports Gear_5	1150	
105	5	Sports Gear_5	1150	
111	6	Ethnic Wear_6	1300	
106	6	Ethnic Wear_6	1300	
112	7	Mobile Phone_7	1450	
107	7	Mobile Phone_7	1450	
113	8	Book_8	1600	
108	8	Book_8	1600	
114	9	Home Appliance_9	1750	
109	9	Home Appliance_9	1750	
115	10	Sports Gear_10	1900	
110	10	Sports Gear_10	1900	
116	11	Ethnic Wear_11	2050	
111	11	Ethnic Wear_11	2050	
117	12	Mobile Phone_12	2200	
112	12	Mobile Phone_12	2200	

-- 4. SUBQUERY: Customers who placed 2 or more orders

SELECT

```
    c.customer_id,
    c.customer_name,
    ( SELECT COUNT(*) FROM orders o WHERE o.customer_id =
c.customer_id ) AS order_count
```

FROM

```
    customers c
```

WHERE

```
    ( SELECT COUNT(*) FROM orders o WHERE o.customer_id =
c.customer_id ) >= 2;
```

Output:

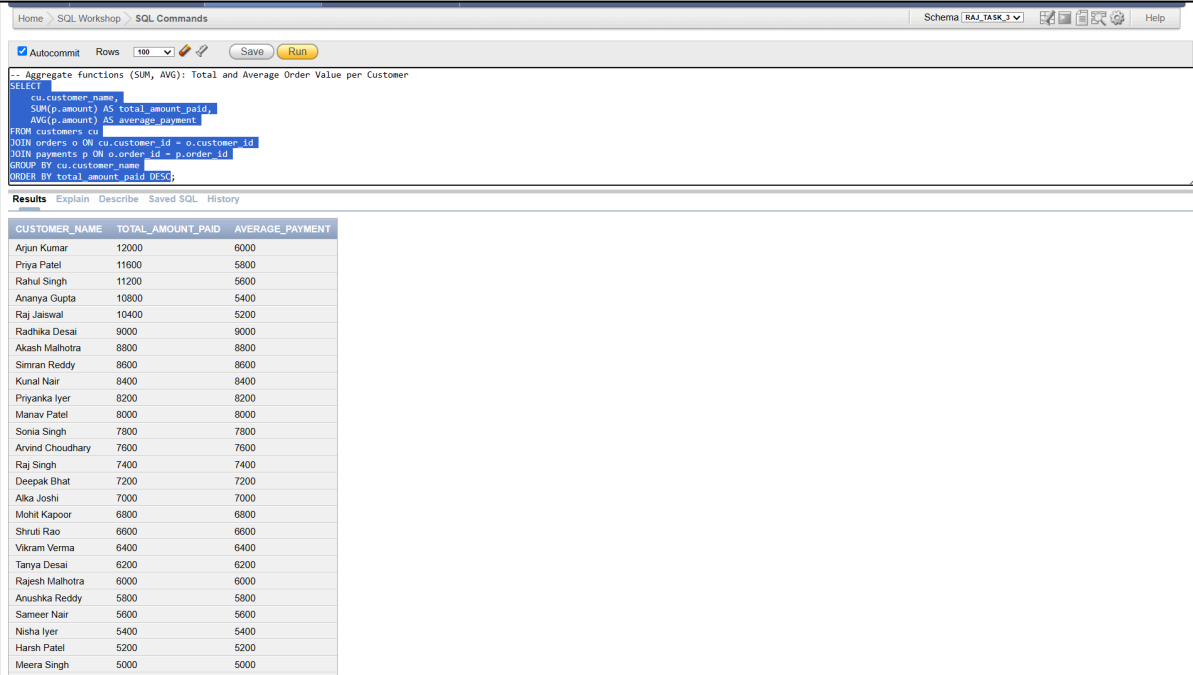
ORACLE Application Express				
Welcome SYSTEM (Logout)				
Home Application Builder SQL Workshop Team Development Administration				
Home > SQL Workshop > SQL Commands				
Schema: RAJ_TASK_3				
Autocommit Rows: 100 Save Run				
<pre>-- SUBQUERY: Customers who placed 2 or more orders SELECT c.customer_id, c.customer_name, (SELECT COUNT(*) FROM orders o WHERE o.customer_id = c.customer_id) AS order_count FROM customers c WHERE (SELECT COUNT(*) FROM orders o WHERE o.customer_id = c.customer_id) >= 2;</pre>				
Results Explain Describe Saved SQL History				
CUSTOMER_ID	CUSTOMER_NAME	ORDER_COUNT		
1	Raj Jaiswal	2		
2	Ananya Gupta	2		
3	Rahul Singh	2		
4	Priya Patel	2		
5	Arjun Kumar	2		

5 rows returned in 0.01 seconds Download

-- 5. Aggregate functions (SUM, AVG): Total and Average Order Value per Customer

```
SELECT
    cu.customer_name,
    SUM(p.amount) AS total_amount_paid,
    AVG(p.amount) AS average_payment
FROM
    customers cu
JOIN
    orders o ON cu.customer_id = o.customer_id
JOIN
    payments p ON o.order_id = p.order_id
GROUP BY
    cu.customer_name
ORDER BY
    total_amount_paid DESC;
```

Output:



The screenshot shows a SQL Workshop interface with the following components:

- SQL Commands Panel:** Contains the SQL query for calculating total and average order values per customer.
- Results Panel:** Displays the output of the query as a table with three columns: CUSTOMER_NAME, TOTAL_AMOUNT_PAID, and AVERAGE_PAYMENT.

CUSTOMER_NAME	TOTAL_AMOUNT_PAID	AVERAGE_PAYMENT
Arjun Kumar	12000	6000
Priya Patel	11600	5800
Rahul Singh	11200	5600
Ananya Gupta	10800	5400
Raj Jaiswal	10400	5200
Radhika Desai	9000	9000
Akash Malhotra	8800	8800
Simran Reddy	8600	8600
Kunal Nair	8400	8400
Priyanka Iyer	8200	8200
Manav Patel	8000	8000
Sonia Singh	7800	7800
Arvind Choudhary	7600	7600
Raj Singh	7400	7400
Deepak Bhat	7200	7200
Alka Joshi	7000	7000
Mohit Kapoor	6800	6800
Shruti Rao	6600	6600
Vikram Verma	6400	6400
Tanya Desai	6200	6200
Rajesh Malhotra	6000	6000
Anushka Reddy	5800	5800
Sameer Nair	5600	5600
Nisha Iyer	5400	5400
Harsh Patel	5200	5200
Meera Singh	5000	5000

-- 6. Create an enhanced view for monthly sales analysis

```
CREATE OR REPLACE VIEW monthly_sales AS
SELECT
    TO_CHAR(o.order_date, 'YYYY-MM') AS month,
    COUNT(DISTINCT o.order_id) AS total_orders,
    COUNT(DISTINCT o.customer_id) AS unique_customers,
    SUM(oi.quantity) AS total_quantity_sold
FROM
    orders o
JOIN
    order_items oi ON o.order_id = oi.order_id
GROUP BY
    TO_CHAR(o.order_date, 'YYYY-MM');

-- Use the view
SELECT * FROM monthly_sales;
```

Output:

The screenshot shows the Oracle Application Express interface. The SQL Commands window contains the SQL script to create the view. The Results window shows the message "View created." and the execution time "0.02 seconds".

```
-- Create an enhanced view for monthly sales analysis
CREATE OR REPLACE VIEW monthly_sales AS
SELECT
    TO_CHAR(o.order_date, 'YYYY-MM') AS month,
    COUNT(DISTINCT o.order_id) AS total_orders,
    COUNT(DISTINCT o.customer_id) AS unique_customers,
    SUM(oi.quantity) AS total_quantity_sold
FROM
    orders o
JOIN
    order_items oi ON o.order_id = oi.order_id
GROUP BY
    TO_CHAR(o.order_date, 'YYYY-MM');

-- Use the view
SELECT * FROM monthly_sales;
```

Results Explain Describe Saved SQL History

View created.

0.02 seconds

The screenshot shows the Oracle Application Express interface. The SQL Commands window contains the SQL script to query the view. The Results window shows a table with 2 rows of data.

```
-- Create an enhanced view for monthly sales analysis
CREATE OR REPLACE VIEW monthly_sales AS
SELECT
    TO_CHAR(o.order_date, 'YYYY-MM') AS month,
    COUNT(DISTINCT o.order_id) AS total_orders,
    COUNT(DISTINCT o.customer_id) AS unique_customers,
    SUM(oi.quantity) AS total_quantity_sold
FROM
    orders o
JOIN
    order_items oi ON o.order_id = oi.order_id
GROUP BY
    TO_CHAR(o.order_date, 'YYYY-MM');

-- Use the view
SELECT * FROM monthly_sales;
```

Results Explain Describe Saved SQL History

MONTH	TOTAL_ORDERS	UNIQUE_CUSTOMERS	TOTAL_QUANTITY_SOLD
2025-01	30	30	165
2025-02	15	15	45

2 rows returned in 0.01 seconds Download

-- 7. Optimize queries with indexes: Create index on frequently searched columns

```
CREATE INDEX idx_products_price ON products(price);
CREATE INDEX idx_orders_customer ON orders(customer_id);
```

```
-- View created indexes
```

```
SELECT
    index_name,
    table_name,
    column_name
FROM
    user_ind_columns
WHERE
    table_name IN ('PRODUCTS', 'ORDERS');
```

Output:

ORACLE Application Express Welcome SYSTEM (Logout)

Home Application Builder SQL Workshop Team Development Administration

Home > SQL Workshop > SQL Commands Schema: RAJ_TASK_3

☒ Autocommit Rows: 100 Save Run

```
-- Create index on frequently searched columns
CREATE INDEX idx_products_price ON products(price);
CREATE INDEX idx_orders_customer ON orders(customer_id);

SELECT index_name, table_name, column_name
FROM user_ind_columns
WHERE table_name IN ('PRODUCTS', 'ORDERS');
```

Results Explain Describe Saved SQL History

Index created.

0.02 seconds

ORACLE Application Express Welcome SYSTEM (Logout)

Home Application Builder SQL Workshop Team Development Administration

Home > SQL Workshop > SQL Commands Schema: RAJ_TASK_3

☒ Autocommit Rows: 100 Save Run

```
-- Create index on frequently searched columns
CREATE INDEX idx_products_price ON products(price);
CREATE INDEX idx_orders_customer ON orders(customer_id);

SELECT index_name, table_name, column_name
FROM user_ind_columns
WHERE table_name IN ('PRODUCTS', 'ORDERS');
```

Results Explain Describe Saved SQL History

INDEX_NAME	TABLE_NAME	COLUMN_NAME
SYS_C007643	ORDERS	ORDER_ID
IDX_ORDERS_CUSTOMER	ORDERS	CUSTOMER_ID
SYS_C007641	PRODUCTS	PRODUCT_ID
IDX_PRODUCTS_PRICE	PRODUCTS	PRICE

4 rows returned in 0.01 seconds [Download](#)