**My sql Queries :**

| **Query type** | **Sql query example** | **Explanation** |
| --- | --- | --- |
| **Select all data** | **Select \* from your\_table\_name;** | Retrieves all columns and rows from the table. |
| **Select specific columns** | **Select merchantname, amount, datetime from your\_table\_name;** | Retrieves specified columns from the table. |
| **Count rows** | **Select count(\*) as total\_rows from your\_table\_name;** | Counts the total number of rows in the table. |
| **Count rows with condition** | **Select count(\*) as success\_count from your\_table\_name where status = 'success';** | Counts rows where status is 'success'. |
| **Sum of amounts** | **Select sum(amount) as total\_amount from your\_table\_name;** | Calculates the total sum of the amount column. |
| **Average commission** | **Select avg(commission) as average\_commission from your\_table\_name;** | Calculates the average value of the commission column. |
| **Total amount by merchant** | **Select merchantname, sum(amount) as total\_amount from your\_table\_name group by merchantname;** | Groups by merchantname and calculates total amount per merchant. |
| **Count of transactions by status** | **Select status, count(\*) as transaction\_count from your\_table\_name group by status;** | Counts transactions grouped by status. |
| **Filter transactions by date** | **Select \* from your\_table\_name where datetime between '2024-01-01' and '2024-12-31';** | Retrieves transactions within a date range. |
| **Find transactions with merchantid** | **Select \* from your\_table\_name where merchantid = 'specific\_id';** | Retrieves transactions for a specific merchantid. |
| **Retrieve transactions with gateway** | **Select \* from your\_table\_name where gateway = 'specific\_gateway';** | Retrieves transactions that use a specific gateway. |
| **Find top 10 transactions by amount** | **Select \* from your\_table\_name order by amount desc limit 10;** | Retrieves the top 10 transactions with the highest amount. |

**Advanced queries table**

| **Query type** | **Sql query example** | **Explanation** |
| --- | --- | --- |
| **Join two tables (inner join)** | **Sql use your\_database\_name; select t1.merchantname, t2.amount from transactions t1 inner join payments t2 on t1.merchantid = t2.merchantid;** | Joins transactions and payments tables on merchantid to retrieve related data. |
| **Left join (all from left table)** | **Sql use your\_database\_name; select t1.merchantname, t2.amount from transactions t1 left join payments t2 on t1.merchantid = t2.merchantid;** | Retrieves all rows from transactions and matching rows from payments. |
| **Right join (all from right table)** | **Sql use your\_database\_name; select t1.merchantname, t2.amount from transactions t1 right join payments t2 on t1.merchantid = t2.merchantid;** | Retrieves all rows from payments and matching rows from transactions. |
| **Full outer join (all from both tables)** | **Sql use your\_database\_name; select t1.merchantname, t2.amount from transactions t1 full outer join payments t2 on t1.merchantid = t2.merchantid;** | Retrieves all rows from both transactions and payments, with null where there is no match. |
| **Self join** | **Sql use your\_database\_name; select a.merchantname as merchant1, b.merchantname as merchant2 from transactions a inner join transactions b on a.merchantid = b.merchantid and a.id <> b.id;** | Joins the transactions table with itself to find pairs of rows with the same merchantid. |
| **Aggregate with join** | **Sql use your\_database\_name; select t1.merchantname, sum(t2.amount) as total\_amount from transactions t1 inner join payments t2 on t1.merchantid = t2.merchantid group by t1.merchantname;** | Aggregates amount for each merchantname after joining transactions and payments. |
| **Subquery with join** | **Sql use your\_database\_name; select merchantname, amount from transactions where merchantid in (select merchantid from payments where amount > 1000);** | Retrieves transactions where the merchantid matches those with amounts greater than 1000 in payments. |
| **Filter joined data** | **Sql use your\_database\_name; select t1.merchantname, t2.amount from transactions t1 inner join payments t2 on t1.merchantid = t2.merchantid where t2.amount > 500;** | Retrieves data where amount in payments is greater than 500, after joining transactions. |
| **Join with aggregation and filtering** | **Sql use your\_database\_name; select t1.merchantname, count(t2.id) as transaction\_count from transactions t1 inner join payments t2 on t1.merchantid = t2.merchantid group by t1.merchantname having count(t2.id) > 5;** | Retrieves merchants with more than 5 transactions by joining transactions and payments |

**Advanced sql queries table**

| **Query type** | **Sql query example** | **Explanation** |
| --- | --- | --- |
| **Window functions** | **Sql use your\_database\_name; select merchantname, amount, row\_number() over (partition by merchantname order by amount desc) as row\_num from transactions;** | Uses row\_number() to assign a unique number to each row within a partition. |
| **Running total** | **Sql use your\_database\_name; select datetime, merchantname, amount, sum(amount) over (partition by merchantname order by datetime) as running\_total from transactions;** | Calculates a running total of amount for each merchant over time. |
| **Rank by value** | **Sql use your\_database\_name; select merchantname, amount, rank() over (order by amount desc) as rank from transactions;** | Ranks merchants based on the amount, with the highest amount receiving rank 1. |
| **Pivot table** | **Sql use your\_database\_name; select merchantname, sum(case when status = 'success' then amount else 0 end) as success\_amount, sum(case when status = 'failure' then amount else 0 end) as failure\_amount from transactions group by merchantname;** | Converts rows into columns to compare different statuses. |
| **Complex join with aggregation** | **Sql use your\_database\_name; select m.merchantname, count(t.id) as transaction\_count, sum(t.amount) as total\_amount from merchants m left join transactions t on m.merchantid = t.merchantid group by m.merchantname;** | Joins merchants and transactions, counts transactions, and sums amounts per merchant. |
| **Hierarchical data (recursive query)** | **Sql use your\_database\_name; with recursive hierarchy as ( select id, parentid, name from categories where parentid is null union all select c.id, c.parentid, c.name from categories c inner join hierarchy h on c.parentid = h.id ) select \* from hierarchy;** | Retrieves hierarchical data, such as organizational charts or category structures. |
| **Time series analysis** | **Sql use your\_database\_name; select date\_trunc('month', datetime) as month, sum(amount) as monthly\_total from transactions group by month order by month;** | Aggregates data by month for time series analysis. |
| **Comparative analysis** | **Sql use your\_database\_name; select t1.merchantname, t1.amount as currentmonthamount, t2.amount as previousmonthamount from transactions t1 left join transactions t2 on t1.merchantid = t2.merchantid and date\_trunc('month', t1.datetime) = date\_trunc('month', t2.datetime) + interval '1 month' where t1.datetime >= '2024-01-01';** | Compares current month amounts with previous month amounts for each merchant. |
| **Data deduplication** | **Sql use your\_database\_name; with deduped as ( select id, merchantname, amount, row\_number() over (partition by merchantid order by datetime desc) as row\_num from transactions ) delete from deduped where row\_num > 1;** | Removes duplicate records based on a specified condition, keeping the most recent. |
| **Custom aggregations** | **Sql use your\_database\_name; select merchantname, count(distinct morderid) as unique\_orders, avg(amount) as average\_amount from transactions group by merchantname;** | Performs custom aggregations, such as counting unique orders and averaging amounts. |

**Commonly used sql queries for data reporting**

| **Query type** | **Sql query example** | **Explanation** |
| --- | --- | --- |
| **Top n performing merchants** | **Sql use your\_database\_name; select merchantname, sum(amount) as total\_amount from transactions group by merchantname order by total\_amount desc limit 10;** | Retrieves the top 10 merchants based on total transaction amount. |
| **Monthly sales report** | **Sql use your\_database\_name; select date\_format(datetime, '%y-%m') as month, sum(amount) as total\_sales from transactions group by month order by month;** | Aggregates sales data by month for reporting purposes. |
| **Year-over-year comparison** | **Sql use your\_database\_name; select year(datetime) as year, sum(amount) as total\_amount from transactions group by year order by year;** | Compares sales or transaction amounts year-over-year. |
| **Sales by region** | **Sql use your\_database\_name; select region, sum(amount) as total\_sales from transactions group by region;** | Aggregates sales data by geographical region or department. |
| **Customer segmentation** | **Sql use your\_database\_name; select customerid, count(\*) as purchase\_count, sum(amount) as total\_spent from transactions group by customerid having total\_spent > 1000;** | Segments customers based on their spending behavior. |
| **Top products by sales** | **Sql use your\_database\_name; select productid, sum(amount) as total\_sales from sales group by productid order by total\_sales desc limit 5;** | Retrieves the top 5 products by sales amount. |
| **Daily sales trend** | **Sql use your\_database\_name; select date(datetime) as day, sum(amount) as daily\_total from transactions group by day order by day;** | Provides a daily breakdown of sales to analyze trends over time. |
| **Sales funnel analysis** | **Sql use your\_database\_name; select stage, count(\*) as count from sales\_funnel group by stage;** | Analyzes the number of deals or transactions at each stage of the sales funnel. |
| **Sales performance by employee** | **Sql use your\_database\_name; select employeeid, sum(salesamount) as total\_sales from sales group by employeeid order by total\_sales desc;** | Evaluates sales performance by employee. |
| **Customer lifetime value** | **Sql use your\_database\_name; select customerid, sum(amount) as lifetime\_value from transactions group by customerid;** | Calculates the total value generated from each customer over their lifetime. |
| **Product return analysis** | **Sql use your\_database\_name; select productid, count(\*) as return\_count from returns group by productid;** | Analyzes the number of returns by product to identify issues. |
| **Sales by channel** | **Sql use your\_database\_name; select sales\_channel, sum(amount) as total\_sales from transactions group by sales\_channel;** | Aggregates sales data by different sales channels or platforms. |
| **Revenue growth rate** | **Sql use your\_database\_name; select year(datetime) as year, (sum(amount) - lag(sum(amount)) over (order by year(datetime))) / lag(sum(amount)) over (order by year(datetime)) \* 100 as growth\_rate from transactions group by year;** | Calculates the year-over-year revenue growth rate. |
| **Cross-selling opportunities** | **Sql use your\_database\_name; select a.customerid, count(distinct b.productid) as cross\_selling\_count from transactions a join transactions b on a.customerid = b.customerid and a.productid <> b.productid group by a.customerid having cross\_selling\_count > 1;** | Identifies customers who purchased multiple different products. |
| **Profit margin analysis** | **Sql use your\_database\_name; select productid, (sum(amount) - sum(cost)) / sum(amount) \* 100 as profit\_margin from sales group by productid;** | Calculates the profit margin for each product. |

**Common sql queries for specific data requests**

| **Request type** | **Sql query example** | **Explanation** |
| --- | --- | --- |
| **Find specific records** | **Sql use your\_database\_name; select \* from transactions where merchantname = 'merchant x';** | Retrieves all records where the merchantname matches 'merchant x'. |
| **Find records with multiple criteria** | **Sql use your\_database\_name; select \* from transactions where amount > 1000 and status = 'success';** | Retrieves records where amount is greater than 1000 and status is 'success'. |
| **Retrieve recent transactions** | **Sql use your\_database\_name; select \* from transactions where datetime >= now() - interval 7 day;** | Retrieves transactions from the last 7 days. |
| **Find transactions for a specific merchant** | **Sql use your\_database\_name; select \* from transactions where merchantid = '12345';** | Retrieves transactions for a specific merchantid. |
| **Total sales for a period** | **Sql use your\_database\_name; select sum(amount) as total\_sales from transactions where datetime between '2024-01-01' and '2024-01-31';** | Calculates the total sales amount within a specific date range. |
| **Top n customers by spend** | **Sql use your\_database\_name; select customerid, sum(amount) as total\_spent from transactions group by customerid order by total\_spent desc limit 10;** | Retrieves the top 10 customers by total spend. |
| **Sales summary by product** | **Sql use your\_database\_name; select productid, sum(amount) as total\_sales from sales group by productid;** | Provides a sales summary for each product. |
| **Find duplicate records** | **Sql use your\_database\_name; select merchantid, count(\*) as count from transactions group by merchantid having count(\*) > 1;** | Identifies duplicate records based on merchantid. |
| **Average transaction amount** | **Sql use your\_database\_name; select avg(amount) as average\_amount from transactions;** | Calculates the average transaction amount across all records. |
| **Sales by region for a specific year** | **Sql use your\_database\_name; select region, sum(amount) as total\_sales from transactions where year(datetime) = 2024 group by region;** | Aggregates sales by region for the year 2024. |
| **Record with maximum value** | **Sql use your\_database\_name; select \* from transactions order by amount desc limit 1;** | Retrieves the record with the highest transaction amount. |
| **Check data integrity** | **Sql use your\_database\_name; select \* from transactions t left join merchants m on t.merchantid = m.merchantid where m.merchantid is null;** | Identifies transactions with missing or unmatched merchant data. |
| **Find transactions by status** | **Sql use your\_database\_name; select \* from transactions where status = 'failure';** | Retrieves transactions that have a status of 'failure'. |
| **Count transactions by date** | **Sql use your\_database\_name; select date(datetime) as day, count(\*) as transaction\_count from transactions group by day;** | Counts the number of transactions per day. |
| **Find transactions with large amounts** | **Sql use your\_database\_name; select \* from transactions where amount > (select avg(amount) from transactions);** | Retrieves transactions with amounts greater than the average transaction amount. |
| **Find records with missing data** | **Sql use your\_database\_name; select \* from transactions where merchantemail is null;** | Retrieves records where merchantemail is missing or null. |
| **Sales by payment gateway** | **Sql use your\_database\_name; select gateway, sum(amount) as total\_sales from transactions group by gateway;** | Aggregates sales by payment gateway. |

**SQL Queries for Data and Schema Management**

| **Operation** | **SQL Query Example** | **Explanation** |
| --- | --- | --- |
| **Delete Data** | Sql USE your\_database\_name; DELETE FROM transactions WHERE Status = 'failure'; | Deletes records from the transactions table where the Status is 'failure'. |
| **Delete All Data** | Sql USE your\_database\_name; DELETE FROM transactions; | Deletes all records from the transactions table but keeps the table structure. |
| **Drop Table** | Sql USE your\_database\_name; DROP TABLE transactions; | Completely removes the transactions table from the database, including all data and structure. |
| **Drop Database** | Sql DROP DATABASE your\_database\_name; | Deletes the entire database, including all tables, data, and objects within it. |
| **Alter Table - Add Column** | Sql USE your\_database\_name; ALTER TABLE transactions ADD COLUMN newcolumn VARCHAR(255); | Adds a new column named newcolumn to the transactions table. |
| **Alter Table - Drop Column** | Sql USE your\_database\_name; ALTER TABLE transactions DROP COLUMN oldcolumn; | Removes an existing column named oldcolumn from the transactions table. |
| **Alter Table - Modify Column** | Sql USE your\_database\_name; ALTER TABLE transactions MODIFY COLUMN Amount DECIMAL(10,2); | Modifies the data type of the Amount column to DECIMAL(10,2) in the transactions table. |
| **Alter Table - Rename Column** | Sql USE your\_database\_name; ALTER TABLE transactions RENAME COLUMN oldname TO newname; | Renames the column oldname to newname in the transactions table. |
| **Alter Table - Add Primary Key** | Sql USE your\_database\_name; ALTER TABLE transactions ADD PRIMARY KEY (transactionid); | Adds a primary key constraint on the transactionid column in the transactions table. |
| **Alter Table - Drop Primary Key** | Sql USE your\_database\_name; ALTER TABLE transactions DROP PRIMARY KEY; | Removes the primary key constraint from the transactions table. |
| **Alter Table - Add Foreign Key** | Sql USE your\_database\_name; ALTER TABLE transactions ADD CONSTRAINT fk\_merchant FOREIGN KEY (merchantid) REFERENCES merchants(merchantid); | Adds a foreign key constraint to the merchantid column, referencing the merchantid in the merchants table. |
| **Alter Table - Drop Foreign Key** | Sql USE your\_database\_name; ALTER TABLE transactions DROP FOREIGN KEY fk\_merchant; | Removes a foreign key constraint named fk\_merchant from the transactions table. |