SQL Notes for Data Analysts

1. What is SQL?

SQL (**Structured Query Language**) is used to **store, retrieve, update, and manage data** in databases. It helps in **analysing and organizing** large datasets efficiently.

---

2. Basic SQL Commands

Command	**Purpose**	**Example**
SELECT	Retrieve data	SELECT * FROM Customers;
INSERT	Add new data	INSERT INTO Customers VALUES (1, 'John', 25);
UPDATE	Modify existing data	UPDATE Customers SET Age = 30 WHERE ID = 1;
DELETE	Remove data	DELETE FROM Customers WHERE ID = 1;

Example:

```
SELECT Name, Age FROM Customers WHERE Age > 25;
```

---

3. Creating and Managing Tables

3.1 Creating a Table (CREATE TABLE)

```
CREATE TABLE Customers (
   ID INT PRIMARY KEY,
   Name VARCHAR(50),
   Age INT
);
```

3.2 Modifying a Table (ALTER TABLE)

- **Add a new column:**

```
ALTER TABLE Customers ADD Email VARCHAR(100);
```

- **Rename a column:**

```
ALTER TABLE Customers RENAME COLUMN Age TO Customer_Age;
```

- **Delete a column:**

```
ALTER TABLE Customers DROP COLUMN Email;
```

3.3 Deleting a Table (DROP TABLE)

```
DROP TABLE Customers;
```

---

4. Filtering Data (WHERE Clause)

```
SELECT * FROM Sales WHERE Revenue > 5000;
```

---

5. Sorting and Limiting Data

5.1 Sorting (ORDER BY)

```
SELECT * FROM Sales ORDER BY Revenue DESC;
```

5.2 Limiting Rows (LIMIT)

```
SELECT * FROM Sales LIMIT 5;
```

---

6. Aggregate Functions

```
SELECT COUNT(*) FROM Customers;
```

```
SELECT AVG(Revenue) FROM Sales;
```

---

7. Grouping Data (GROUP BY & HAVING)

```
SELECT Region, SUM(Revenue) FROM Sales GROUP BY Region;
```

```
SELECT Region, COUNT(*) FROM Sales GROUP BY Region HAVING COUNT(*) > 10;
```

---

8. Joining Tables (JOIN)

8.1 INNER JOIN

```
SELECT Customers.Name, Orders.Order_ID FROM Customers INNER JOIN Orders ON
Customers.ID = Orders.Customer_ID;
```

8.2 LEFT JOIN

```
SELECT Customers.Name, Orders.Order_ID FROM Customers LEFT JOIN Orders ON
Customers.ID = Orders.Customer_ID;
```

---

9. Constraints in SQL

Constraint	**Purpose**
PRIMARY KEY	Ensures uniqueness
FOREIGN KEY	Links tables
NOT NULL	Prevents NULL values
UNIQUE	Ensures unique values
CHECK	Sets conditions
DEFAULT	Assigns default values

---

10. Views (CREATE VIEW)

```
CREATE VIEW HighRevenue AS SELECT * FROM Sales WHERE Revenue > 10000;
```

---

11. Indexing (CREATE INDEX)

```
CREATE INDEX idx_name ON Customers(Name);
```

---

12. Transactions (START TRANSACTION)

```
START TRANSACTION;

UPDATE Sales SET Revenue = Revenue + 1000 WHERE Region = 'East';

COMMIT; -- Save changes

ROLLBACK; -- Undo changes
```

---

13. Truncating Data (TRUNCATE)

```
TRUNCATE TABLE Sales;
```

---

14. Window Functions

14.1 Ranking Functions (ROW_NUMBER, RANK, DENSE_RANK)

```
SELECT Employee_ID, Name, Department,
    ROW_NUMBER() OVER (PARTITION BY Department ORDER BY Salary DESC) AS RowNum
FROM Employees;
```

14.2 LAG() and LEAD()

```
SELECT Month, Sales,
    LAG(Sales) OVER (ORDER BY Month) AS PreviousMonthSales,
    LEAD(Sales) OVER (ORDER BY Month) AS NextMonthSales
FROM SalesData;
```

14.3 Running Total (SUM() OVER)

```
SELECT Employee_ID, Name, Salary,
    SUM(Salary) OVER (ORDER BY Employee_ID) AS RunningTotal
FROM Employees;
```

📌 Summary Table for Data Analysts

Topic	**Key SQL Command**
Basic Queries	SELECT, WHERE, ORDER BY, LIMIT
Table Operations	CREATE TABLE, ALTER TABLE, DROP TABLE
Data Modification	INSERT, UPDATE, DELETE, TRUNCATE
Filtering & Aggregation	GROUP BY, HAVING, SUM(), AVG()
Joins & Relationships	INNER JOIN, LEFT JOIN, FOREIGN KEY
Constraints	PRIMARY KEY, NOT NULL, UNIQUE, CHECK
Performance Optimization	INDEX, VIEW, TRANSACTION, TRIGGER
Window Functions	ROW_NUMBER(), LAG(), LEAD(), SUM() OVER

Final Notes

- SQL is essential for a **Data Analyst** to **query, filter, and analyze data** efficiently.
- **Practice with real-world datasets** to improve SQL skills.
- **Learn Joins, Aggregations, and Window Functions** for advanced analytics.
