SQL (Standard Query Language)

1. SELECT

```
-- return all columns and rows
SELECT * FROM Countries;

-- return specific columns
SELECT id, countryName, capital FROM Countries;

-- return unique continent values
SELECT DISTINCT continent FROM Countries;

-- return language for UK only
SELECT lang FROM countries WHERE countryName='UK';

-- return countries in Asia
SELECT * FROM countries WHERE continent='Asia';

-- sort countries by name descending
SELECT * FROM Countries ORDER BY countryName DESC;

-- return capitals sorted alphabetically
SELECT capital FROM Countries ORDER BY capital ASC;

-- pagination: skip 2 rows, return next 5
SELECT * FROM Countries LIMIT 5 OFFSET 2;

-- count countries per continent
SELECT continent, COUNT(*)
FROM Countries
GROUP BY continent;

-- generic: return duplicated values only
SELECT col, COUNT(*) FROM table GROUP BY col HAVING COUNT(*) > 1;

-- return continents with at least 2 countries
SELECT continent, COUNT(*)
FROM Countries GROUP BY continent HAVING COUNT(*)>=2;

-- basic inner join template
SELECT * FROM A INNER JOIN B ON condition;

-- wrong join: filter only, no relationship
SELECT * FROM countries INNER JOIN faith ON religion='Islam';

-- correct inner join with filter
```

```
SELECT * FROM
countries c INNER JOIN faith f
ON c.faithId=f.faithId
WHERE f.religion = 'No Religion';

-- basic left join template
SELECT * FROM A LEFT JOIN B ON condition;

-- left join but WHERE turns it into inner join
SELECT * FROM
countries c LEFT JOIN faith f
ON c.faithId = f.faithId
WHERE f.religion = 'Islam';

-- basic right join template
SELECT * FROM A RIGHT JOIN B ON condition;

-- right join filtered by Christian faith
SELECT * FROM
countries c RIGHT JOIN faith f
ON c.faithId = f.faithId
WHERE f.religion = 'Christian';

-- full join template
-- SELECT * FROM A FULL JOIN B ON condition;
SELECT * FROM
countries c FULL JOIN faith f
ON c.faithId = f.faithId
WHERE f.religion = 'Hindu';

-- right join filtered by Hindu faith
SELECT * FROM
countries c RIGHT JOIN faith f
ON c.faithId = f.faithId
WHERE f.religion = 'Hindu';

-- IN subquery: match values from subquery
SELECT * FROM
countries
WHERE faithId IN(
        SELECT faithId
        FROM faith
        WHERE religion = 'No Religion'
);
```

```
-- EXISTS: returns all rows if Asia exists at least once
SELECT * FROM countries c
WHERE EXISTS(
        SELECT 1
        FROM countries c2
        WHERE c2.continent = 'Asia'
);


-- SELECT col,
--       (SELECT MAX(col) FROM table) AS max_col
-- FROM table;
-- SELECT col FROM table
-- UNION
-- SELECT col FROM table2;

-- SELECT col FROM table
-- UNION ALL
-- SELECT col FROM table2;


-- SELECT col,
--       CASE WHEN condition THEN value ELSE value END
-- FROM table;
```

2. SELECT DISTINCT
3. WHERE
4. ORDER BY
5. AND
6. OR
7. NOT

8. INSERT INTO

Single INSERT INTO

```
INSERT INTO Countries(countryid, countryname, capital, continent, speakinglanguage)
VALUES(1, 'UK', 'London', 'Europe', 'English');
```

Multiple INSERT INTO

```
INSERT INTO Countries(countryid, countryname, capital, continent, speakinglanguage)
VALUES
(2, 'France', 'Paris', 'Europe', 'French'),
(3, 'Japan', 'Tokyo', 'Asia', 'Japanese');
```

NULL VALUES

UPDATE

```
UPDATE Countries
SET continent = ‘Asia’
WHERE name = ‘Bangladesh’;
```

DELETE
SELECT TOP
AGGREGATE FUNCTION
MIN, MAX
COUNT
SUM

SET

UPDATE Countries
SET lang = ‘Bengali’
WHERE id = 5;

AVG
LIKE
WILDCARDS
IN
BETWEEN
ALIASES
JOINS

INNER JOIN

```
SELECT * FROM
countries c INNER JOIN faith f
ON c.faithId=f.faithId
WHERE f.religion = 'Islam';
```

LEFT JOIN
RIGHT JOIN
FULL JOIN
SELF JOIN
UNION
UNION ALL
GROUP BY
HAVING

EXISTS (returns true when at least one match is found, more like OR gate)

```
SELECT * FROM countries c
WHERE EXISTS(
        SELECT 1
        FROM countries c2
        WHERE c2.continent = 'Asia'
);
```

ANY, ALL

SELECT INTO
INSERT INTO SELECT
CASE
NULL FUNCTIONS
COMMENTS
OPERATORS

CREATE DB
CREATE DATABASE testdb OWNER sujon;

DROP DB
BACKUP DB

CREATE TABLE
CREATE TABLE Countries (
countryId int PRIMARY KEY,
countryName VARCHAR(255),
capital VARCHAR(255),
continent VARCHAR(255),
speakingLanguage VARCHAR(255)
);

CREATE TABLE Faith (
faithId int PRIMARY KEY,
religion VARCHAR(255) UNIQUE
)

ADD
ALTER TABLE countries
ADD CONSTRAINT fk_countries_religion
FOREIGN KEY (faithid) REFEREnCES faith(id);

ALTER TABLE
Alter constraint type of the field:
ALTER TABLE Countries
ADD CONSTRAINT unique_country_name UNIQUE (countryName);

Alter table column name:
ALTER TABLE Countries
RENAME COLUMN speakingLanguage TO lang;

ALTER TABLE Countries
ADD COLUMN faith_id INT;

CONSTRAINTS
ALTER TABLE countries
ADD CONSTRAINT fk_countries_religion
FOREIGN KEY (faithId) REFERENCES faith(faithid);

NOT NULL
UNIQUE

PRIMARY KEY
FOREIGN KEY
CHECK
DEFAULT
INDEX
AUTO INCREMENT
DATES
VIEWS
INJECTION
HOSTING
DATA TYPES
LIMIT
OFFSET (SKIP)
Keyset pagination
OFFSET
Keyset pagination
DISTINCT ON
Subqueries
Correlated subqueries
Common Table Expressions WITH
Recursive CTE
INTERSECT
EXCEPT
Window functions
OVER clause
PARTITION BY
ROW_NUMBER
RANK
DENSE_RANK
LAG
LEAD
Running totals
Top-N per group queries
Aggregate FILTER
NULL semantics three-valued logic
IS NULL vs = NULL
COALESCE
NULLIF
Transactions BEGIN COMMIT ROLLBACK
Transaction isolation levels
READ COMMITTED

REPEATABLE READ
SERIALIZABLE
SELECT FOR UPDATE
Deadlocks
Retry logic for transactions
EXPLAIN
EXPLAIN ANALYZE
Sequential scan vs index scan
Composite indexes
Partial indexes
Index ordering
Sargable predicates
Foreign key cascading rules
JSON / JSONB
JSON operators and indexes
Arrays and array operators
UNNEST
Full-text search
GIN and GiST indexes
Generated columns
Sequences
IDENTITY columns
Functions
Stored procedures
Triggers
BEFORE vs AFTER triggers
Constraint triggers
Roles and users
GRANT / REVOKE
Row Level Security
Policies
Advisory locks
Explicit table locks
VACUUM
ANALYZE
Autovacuum behavior
Temporary tables
UNLOGGED tables
Extensions
pg_stat views
Query planning vs execution
Replication basics
Backup and restore strategy

PostgreSQL Setup:

sudo apt update
sudo apt install postgresql postgresql-contrib
psql --version
stop service: sudo systemctl stop postgresql
disable: sudo systemctl disable postgresql
enable postgre: sudo systemctl enable postgresql
check status: sudo systemctl status postgresql
start when using: sudo systemctl start postgresql
connect: sudo -u postgres psql
exit psql: \q

pgadmin4 setup
install
sudo apt update
sudo apt install curl ca-certificates gnupg
add pgadmin gpg key
curl -fsS https://www.pgadmin.org/static/packages_pgadmin_org.pub | sudo gpg --dearmor -o /usr/share/keyrings/pgadmin-keyring.gpg
add repo
echo "deb [signed-by=/usr/share/keyrings/pgadmin-keyring.gpg] https://ftp.postgresql.org/pub/pgadmin/pgadmin4/apt/$(lsb_release -cs) pgadmin4 main" | sudo tee /etc/apt/sources.list.d/pgadmin4.list
sudo apt update
sudo apt install pgadmin4-desktop
pgadmin4

User name setup
open postgre terminal with user: sudo -u postgres psql
create a db: CREATE DATABASE testdb OWNER sujon;

in admin ui:
local host: Local PostgreSQL
port: 5432
name of db: testdb
user: sujon
password:
hostname/address: 127.0.0.1

check db owner name:
SELECT datname, datdba::regrole AS owner
FROM pg_database
WHERE datname = 'testdb2';

make myself owner of db:
ALTER DATABASE testdb2 OWNER TO sujon;

Run as postgre owner and drop a table by force:
sudo -u postgres psql
DROP DATABASE testdb WITH (FORCE);