**Keys in Database:**

Great Talha. Here's a clean and interview-focused breakdown of **all major database keys** using your **ride-sharing app** use case + short examples + ER-style table layout (human-style, not AI-ish 😄).

**✅ 1. Super Key**

Any set of attributes that **uniquely identifies** a row (can have extra unnecessary columns).

🧾 **Example**:  
In Drivers(driver\_id, name, email)

* {driver\_id} ✅ (minimal)
* {driver\_id, email} ✅ (super key but **not minimal**)

**✅ 2. Candidate Key**

**Minimal** set of attributes that uniquely identify a record.

🧾 **Example** in Passengers:

| passenger\_id | phone | email |

|--------------|-----------|------------------|

| P01 | 123456789 | talha@mail.com |

* Candidate Keys: {passenger\_id}, {email}, {phone}
* Only one becomes the **primary key**, rest become **alternate keys**

**✅ 3. Primary Key**

Chosen **candidate key** that uniquely identifies rows and is **not null**

CREATE TABLE Passengers (

passenger\_id VARCHAR PRIMARY KEY,

phone VARCHAR UNIQUE,

email VARCHAR UNIQUE

);

✅ passenger\_id is the **Primary Key**

**✅ 4. Alternate Key**

Other candidate keys that were **not chosen** as the primary key

🧾 In Passengers table:

* email, phone are **alternate keys** (also enforced via UNIQUE)

**✅ 5. Composite Primary Key**

Primary key made from **2 or more columns**

CREATE TABLE RideAssignments (

driver\_id VARCHAR,

ride\_id INT,

PRIMARY KEY(driver\_id, ride\_id)

);

* driver\_id + ride\_id = uniquely identifies assignment  
  ✅ Composite Key

**✅ 6. Unique Key**

Ensures **uniqueness**, but **can be null**

CREATE TABLE PromoCodes (

code\_id SERIAL PRIMARY KEY,

promo\_code VARCHAR UNIQUE,

description TEXT

);

* promo\_code is a **unique key** — no duplicates allowed  
  ❗ But can be NULL

**✅ 7. Foreign Key**

Refers to a **primary key** in another table

CREATE TABLE Rides (

ride\_id INT PRIMARY KEY,

driver\_id VARCHAR,

FOREIGN KEY(driver\_id) REFERENCES Drivers(driver\_id)

);

* driver\_id is a **foreign key**

**✅ 8. Compound Key**

Any key that consists of **multiple columns** (can be PK or not)

🧾 Same as **Composite**, but may not be a PK.

CREATE TABLE PassengerFeedback (

ride\_id INT,

passenger\_id VARCHAR,

feedback TEXT,

-- Not declared PK but still unique

UNIQUE(ride\_id, passenger\_id)

);

**✅ 9. Surrogate Key**

**Artificial, system-generated key** (not from real-world data)

CREATE TABLE Trips (

trip\_id SERIAL PRIMARY KEY,

from\_city TEXT,

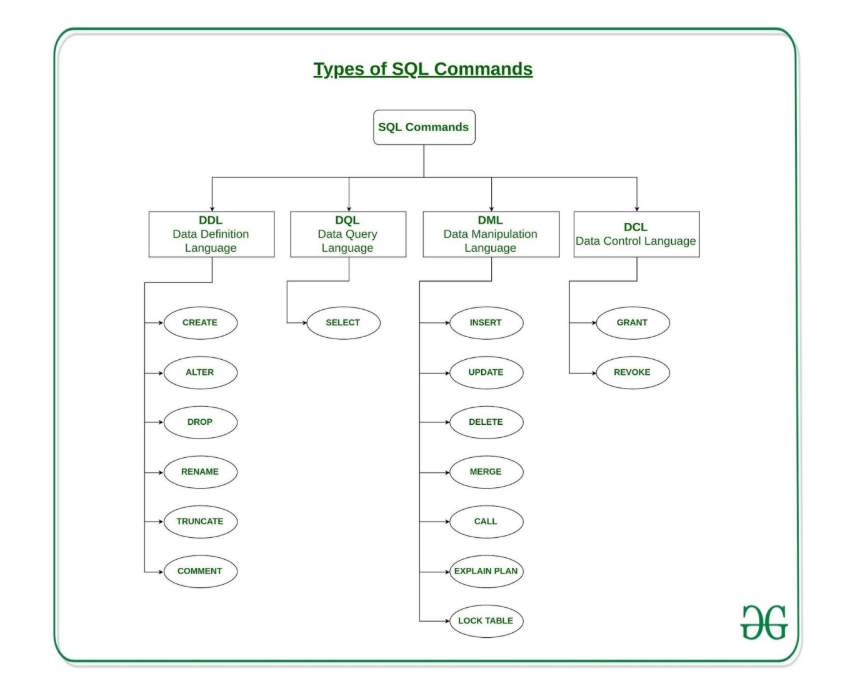
to\_city TEXT

);

* trip\_id is a **surrogate key** (auto-generated by DB)

**👀 Summary Table**

| **Key Type** | **Description** | **Example Table Column** |
| --- | --- | --- |
| Super Key | Any set that uniquely identifies rows | driver\_id, email |
| Candidate Key | Minimal super keys | driver\_id, email |
| Primary Key | Chosen candidate key | passenger\_id |
| Alternate Key | Not-chosen candidate keys | email, phone |
| Composite Primary Key | PK using 2+ fields | driver\_id + ride\_id in assignments |
| Unique Key | Unique but nullable | promo\_code |
| Foreign Key | Links to another table’s PK | driver\_id in Rides table |
| Compound Key | 2+ fields combo (not always PK) | ride\_id + passenger\_id |
| Surrogate Key | Auto-generated system ID | trip\_id (SERIAL or UUID) |



Perfect Talha 👊 — let's make SQL crystal clear using your **ride-sharing app**. We’ll categorize all SQL commands with **real examples**, then wrap up with the **difference between TRUNCATE, DELETE, DROP** using ride-sharing data. 💯

**🔧 1. DDL – Data Definition Language**

*Used to* ***define or alter*** *the structure of tables.*

| **Command** | **Purpose** | **Example (Ride-Sharing)** |
| --- | --- | --- |
| CREATE | Create table | CREATE TABLE Drivers (...) |
| ALTER | Modify table | ALTER TABLE Drivers ADD license\_no VARCHAR; |
| DROP | Delete table | DROP TABLE Passengers; |
| TRUNCATE | Remove all data (keep structure) | TRUNCATE TABLE Rides; |
| RENAME | Rename table | RENAME TABLE Trips TO Journey; |

**🔍 2. DQL – Data Query Language**

*Used to* ***fetch/query*** *data from tables.*

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| SELECT | Read/query data | SELECT \* FROM Drivers WHERE city = 'Lahore'; |

**🛠️ 3. DML – Data Manipulation Language**

*Used to* ***modify data inside*** *tables.*

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| INSERT | Add data | INSERT INTO Passengers VALUES ('P01', 'Talha', '123456789'); |
| UPDATE | Modify data | UPDATE Drivers SET rating = 4.9 WHERE driver\_id = 'D01'; |
| DELETE | Remove specific rows | DELETE FROM Rides WHERE ride\_id = 2; |

**🔐 4. DCL – Data Control Language**

*Used to* ***control access/permissions*** *to data.*

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| GRANT | Give permission | GRANT SELECT ON Rides TO analytics\_user; |
| REVOKE | Take back rights | REVOKE SELECT ON Rides FROM analytics\_user; |

**⚔️ TRUNCATE vs DELETE vs DROP (with data)**

Let’s assume we have:

**🚗 Table: Rides**

ride\_id | passenger\_id | driver\_id | fare

--------|--------------|-----------|------

1 | P01 | D01 | 250

2 | P02 | D02 | 300

**🔥 DELETE FROM Rides WHERE ride\_id = 1;**

* Deletes **1 row**
* Table & structure remain
* Can be **rolled back**
* ✅ Keeps table

**⚡ TRUNCATE TABLE Rides;**

* Deletes **all rows**
* Table remains
* ❌ Cannot rollback (no WHERE)
* ✅ Faster than DELETE

**💣 DROP TABLE Rides;**

* Deletes **table + data**
* ❌ Cannot be rolled back
* ❌ Table is **gone**