1. SELECT CustomerName, City FROM Customers;
2. SELECT DISTINCT Country FROM Customers;
3. SELECT \* FROM Customers WHERE Country='Mexico';
4. SELECT \* FROM Products  
   ORDER BY Price;
5. SELECT \*  
   FROM Customers  
   WHERE Country = 'Spain' AND CustomerName LIKE 'G%';
6. SELECT \*  
   FROM Customers  
   WHERE Country = 'Germany' OR Country = 'Spain';
7. SELECT \* FROM Customers  
   WHERE NOT Country = 'Spain';
8. INSERT INTO *table\_name* (*column1*,*column2*,*column3*, ...)  
   VALUES (*value1*,*value2*,*value3*, ...);
9. SELECT *column\_names*FROM *table\_name*  
   WHERE *column\_name* IS NULL;
10. UPDATE *table\_name*  
    SET *column1*=*value1*,*column2*=*value2*, ...  
    WHERE *condition*;
11. DELETE FROM *table\_name*WHERE *condition*;
12. SELECT TOP 3 \* FROM Customers;
13. MIN() - returns the smallest value within the selected column
14. MAX() - returns the largest value within the selected column
15. COUNT() - returns the number of rows in a set
16. SUM() - returns the total sum of a numerical column
17. AVG() - returns the average value of a numerical column
18. SELECT MIN(Price)  
    FROM Products;
19. SELECT \* FROM Customers  
    WHERE CustomerName LIKE 'a%';

|  |  |
| --- | --- |
| 1. % | 1. Represents zero or more characters |
| 1. \_ | 1. Represents a single character |
| 1. [] | 1. Represents any single character within the brackets \* |
| 1. ^ | 1. Represents any character not in the brackets \* |
| 1. - | 1. Represents any single character within the specified range \* |
| 1. {} | 1. Represents any escaped character \*\* |

1. SELECT \* FROM Customers  
   WHERE Country IN ('Germany', 'France', 'UK');
2. SELECT \* FROM Products  
   WHERE Price BETWEEN 10 AND 20;
3. SELECT CustomerID AS ID  
   FROM Customers;
4. SELECT ProductID, ProductName, CategoryName  
   FROM Products  
   INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;
5. SELECT *column\_name(s)*  
   FROM *table1*  
   LEFT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;
6. SELECT *column\_name(s)*  
   FROM *table1*  
   RIGHT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;
7. SELECT *column\_name(s)*  
   FROM *table1*  
   FULL OUTER JOIN *table2*ON *table1.column\_name*=*table2.column\_name*WHERE *condition*;
8. SELECT *column\_name(s)* FROM *table1*  
   UNION  
   SELECT *column\_name(s)* FROM *table2*;
9. SELECT *column\_name(s)* FROM *table1*  
   UNION ALL  
   SELECT *column\_name(s)* FROM *table2*;
10. SELECT *column\_name(s)*  
    FROM *table\_name*  
    WHERE *condition*  
    GROUP BY *column\_name(s)*ORDER BY *column\_name(s);*
11. SELECT *column\_name(s)*  
    FROM *table\_name*  
    WHERE *condition*  
    GROUP BY *column\_name(s)*HAVING *condition*ORDER BY *column\_name(s);*
12. SELECT *column\_name(s)*  
    FROM *table\_name*  
    WHERE EXISTS  
    (SELECT *column\_name*FROM *table\_name* WHERE *condition*);
13. SELECT *column\_name(s)*  
    FROM *table\_name*  
    WHERE *column\_name operator* ANY  
      (SELECT *column\_name*FROM *table\_name*WHERE *condition*);

Perfect 🎉 let’s make this into a **structured, story-driven adventure** that maps your SQL query list into a full “tour across Sri Lanka.”

I’ll write it in a **game script style** — with **story, character dialogue, tasks, and SQL queries** for each province/level. That way you can directly adapt it into your game engine.

**🗺️ SQL Travels – Full Narrative**

**Level 1 – Western Province (Colombo)**

**Story Intro**  
Alex lands at Bandaranaike International Airport. Excited to start the journey, Alex checks into a hotel in Colombo. Ravi (the driver) and Nila (the local guide) join the adventure.

**Dialogue**

* Ravi: *“Welcome to Sri Lanka! Colombo is busy — let’s find you a hotel first.”*
* Nila: *“This is a great chance to practice your first queries.”*

**Tasks (SQL Basics)**

1. Show all hotels.
2. SELECT \* FROM Hotels;
3. See tourist names and their city.
4. SELECT CustomerName, City FROM Customers;
5. List all nationalities of tourists.
6. SELECT DISTINCT Country FROM Customers;

**Reward**: Badge “Query Beginner” 🏅  
**Unlocked Next**: Central Province.

**Level 2 – Central Province (Kandy)**

**Story Intro**  
Alex takes the famous blue train to Kandy. At Kandy Lake, Alex looks for affordable hotels and meets tourists from different countries.

**Dialogue**

* Nila: *“Kandy is lovely, but your budget is limited. Let’s filter options with conditions.”*
* Ravi: *“Don’t forget to check who else is around!”*

**Tasks (WHERE + Logical Operators)**

1. Find tourists from Mexico.
2. SELECT \* FROM Customers WHERE Country='Mexico';
3. Find hotels in Kandy under Rs. 5000.
4. SELECT \* FROM Hotels WHERE City='Kandy' AND Price < 5000;
5. Find Spanish tourists whose names start with G.
6. SELECT \* FROM Customers WHERE Country='Spain' AND CustomerName LIKE 'G%';
7. Show tourists from Germany or Spain.
8. SELECT \* FROM Customers WHERE Country='Germany' OR Country='Spain';

**Reward**: “Filter Master” Badge 🔍  
**Unlocked Next**: Southern Province.

**Level 3 – Southern Province (Galle)**

**Story Intro**  
Alex visits Galle Fort and the beaches of Unawatuna. Sorting is key — from hotel prices to souvenirs.

**Dialogue**

* Nila: *“Everything here has a price… let’s sort them.”*
* Ravi: *“Surf’s up! But first, can you rank the hotels?”*

**Tasks (ORDER BY + NOT)**

1. Sort products by price.
2. SELECT \* FROM Products ORDER BY Price;
3. Show the top 3 tourists.
4. SELECT TOP 3 \* FROM Customers;
5. Exclude tourists from Spain.
6. SELECT \* FROM Customers WHERE NOT Country = 'Spain';

**Reward**: “Sorting Explorer” Badge 📊  
**Unlocked Next**: Northern Province.

**Level 4 – Northern Province (Jaffna)**

**Story Intro**  
In Jaffna, Alex visits Nallur Kovil. Professor Senanayake appears, teaching how to combine datasets.

**Dialogue**

* Professor: *“Databases are like relationships — everything connects.”*
* Nila: *“Let’s practice JOINs!”*

**Tasks (JOINs)**

1. Show products with category names (INNER JOIN).
2. SELECT ProductID, ProductName, CategoryName
3. FROM Products
4. INNER JOIN Categories ON Products.CategoryID = Categories.CategoryID;
5. Show all tourists, even if no booking (LEFT JOIN).
6. SELECT t.name, b.booking\_id
7. FROM Tourists t
8. LEFT JOIN Bookings b ON t.tourist\_id = b.tourist\_id;
9. Show all bookings, even if no matching tourist (RIGHT JOIN).
10. SELECT t.name, b.booking\_id
11. FROM Tourists t
12. RIGHT JOIN Bookings b ON t.tourist\_id = b.tourist\_id;
13. Bonus: Merge two lists (UNION).
14. SELECT Country FROM Customers
15. UNION
16. SELECT City FROM Hotels;

**Reward**: “Data Connector” Badge 🔗  
**Unlocked Next**: Eastern Province.

**Level 5 – Eastern Province (Batticaloa / Pasikuda)**

**Story Intro**  
Relaxing on Pasikuda beach, Alex helps analyze tourist statistics.

**Dialogue**

* Nila: *“We need some statistics — can you crunch the numbers?”*

**Tasks (Aggregation)**

1. Count tourists per country.
2. SELECT Country, COUNT(\*) FROM Customers GROUP BY Country;
3. Find minimum product price.
4. SELECT MIN(Price) FROM Products;
5. Show hotels priced between 4000 and 6000.
6. SELECT \* FROM Hotels WHERE Price BETWEEN 4000 AND 6000;

**Reward**: “Data Analyzer” Badge 📈  
**Unlocked Next**: North Central Province.

**Level 6 – North Central Province (Anuradhapura)**

**Story Intro**  
At the ancient stupas, Alex encounters a temple puzzle requiring advanced filtering.

**Dialogue**

* Professor: *“WHERE works on rows, HAVING works on groups.”*

**Tasks (GROUP + HAVING)**

1. Cities with more than 2 hotels.
2. SELECT City, COUNT(\*) FROM Hotels GROUP BY City HAVING COUNT(\*) > 2;
3. Tourists who made more than 1 booking.
4. SELECT tourist\_id, COUNT(\*) FROM Bookings GROUP BY tourist\_id HAVING COUNT(\*) > 1;

**Reward**: “Group Master” Badge 🏛️  
**Unlocked Next**: Uva Province.

**Level 7 – Uva Province (Badulla)**

**Story Intro**  
On the Nine Arches Bridge, Alex faces hidden mysteries requiring subqueries.

**Dialogue**

* Nila: *“Subqueries unlock hidden knowledge.”*

**Tasks (Subqueries + EXISTS)**

1. Hotels cheaper than average.
2. SELECT \* FROM Hotels WHERE Price < (SELECT AVG(Price) FROM Hotels);
3. Show tourists who have at least one booking.
4. SELECT \* FROM Tourists WHERE EXISTS (SELECT \* FROM Bookings WHERE Bookings.tourist\_id = Tourists.tourist\_id);

**Reward**: “Temple Solver” Badge 🗝️  
**Unlocked Next**: Sabaragamuwa Province.

**Level 8 – Sabaragamuwa Province (Ratnapura)**

**Story Intro**  
In the gem capital, Alex practices changing data records.

**Dialogue**

* Ravi: *“Like polishing gems, you can update the database!”*

**Tasks (DML)**

1. Add a new hotel.
2. INSERT INTO Hotels (name, city, province\_id, price, amenities) VALUES ('Gemstone Hotel', 'Ratnapura', 8, 5000, 'WiFi, Pool');
3. Update hotel price.
4. UPDATE Hotels SET price = 6000 WHERE name='Gemstone Hotel';
5. Delete duplicates.
6. DELETE FROM Hotels WHERE hotel\_id=12;

**Reward**: “Database Operator” Badge 💎  
**Unlocked Next**: North Western Province.

**Level 9 – North Western Province (Kurunegala)**

**Final Boss Battle**  
At Athugala Rock, Professor Senanayake presents the final exam. Alex must combine everything learned.

**Tasks (Comprehensive)**

1. Create a view for all hotels with WiFi.
2. CREATE VIEW HotelsWithWiFi AS
3. SELECT \* FROM Hotels WHERE amenities LIKE '%WiFi%';
4. Generate a report combining tourists + hotels + bookings.
5. SELECT t.name, h.name AS hotel, b.check\_in
6. FROM Tourists t
7. JOIN Bookings b ON t.tourist\_id = b.tourist\_id
8. JOIN Hotels h ON b.hotel\_id = h.hotel\_id;
9. Find tourists who stayed in more than 1 province.
10. SELECT t.name, COUNT(DISTINCT h.province\_id) AS provinces
11. FROM Tourists t
12. JOIN Bookings b ON t.tourist\_id = b.tourist\_id
13. JOIN Hotels h ON b.hotel\_id = h.hotel\_id
14. GROUP BY t.name
15. HAVING COUNT(DISTINCT h.province\_id) > 1;

**Ending Story**  
Alex is crowned *“SQL Master Explorer.”* Fireworks light up the sky over Sri Lanka. Ravi and Nila cheer, and Professor Senanayake hands over a **certificate of completion**.

**Game Ends** 🎉

👉 This gives you a **full, organized story + matching SQL progression**.  
Would you like me to now **convert this into a level table (SQL INSERTs for your levels table)** so you can load all these tasks directly into your game database?