

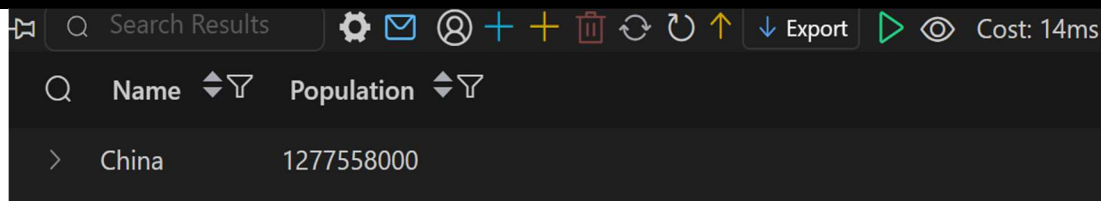
# DPP Session 23 – SQL FUNCTIONS

## ASSIGNMENT

**Question 1: Write an SQL query to find the country with the maximum population in the country table.**

SQL Query:

```
SELECT Name, Population FROM country ORDER BY Population DESC LIMIT 1;
```



The screenshot shows a database interface with a search bar and various icons. Below the search bar, there is a table with two columns: 'Name' and 'Population'. The table contains one row with the value 'China' under 'Name' and '1277558000' under 'Population'.

Name	Population
China	1277558000

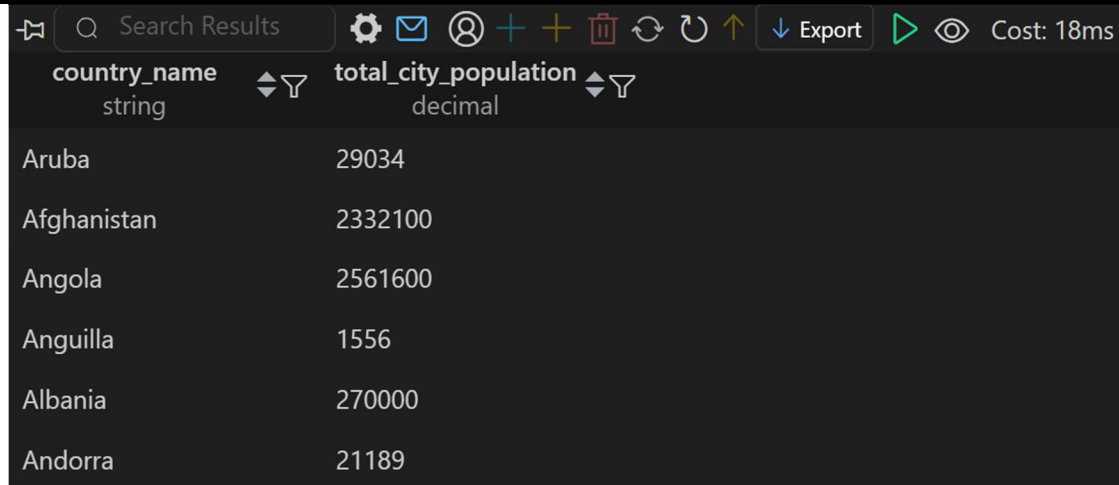
Explanation:

We sort countries by population in descending order and pick the top record.

**Question 2: Write an SQL query to sum the populations of all cities per country.**

SQL Query:

```
SELECT c.Name AS country_name, SUM(ci.Population) AS total_city_population  
FROM country c  
JOIN city ci ON c.Code = ci.CountryCode  
GROUP BY c.Name;
```



The screenshot shows a database interface with a search bar and various icons. Below the search bar, there is a table with two columns: 'country\_name' and 'total\_city\_population'. The table contains six rows of data.

country_name string	total_city_population decimal
Aruba	29034
Afghanistan	2332100
Angola	2561600
Anguilla	1556
Albania	270000
Andorra	21189

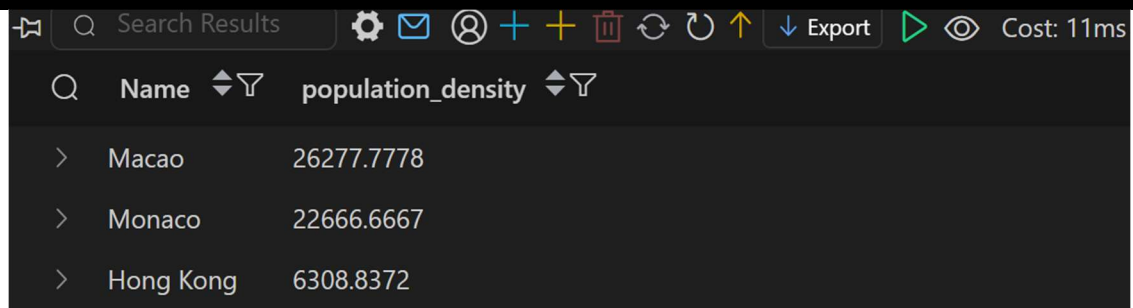
Explanation:

We join country and city tables and aggregate city populations per country.

**Question 3: Find the top 3 countries with the highest population density.**

SQL Query:

```
SELECT Name, (Population / SurfaceArea) AS population_density
FROM country
WHERE SurfaceArea > 0
ORDER BY population_density DESC
LIMIT 3;
```



The screenshot shows a database interface with a search bar and various icons. Below the search bar, the results of the SQL query are displayed in a table. The table has two columns: 'Name' and 'population\_density'. The results are sorted in descending order of population density. The top three results are Macao, Monaco, and Hong Kong.

Name	population_density
Macao	26277.7778
Monaco	22666.6667
Hong Kong	6308.8372

Explanation:

Population density is calculated as population divided by surface area.

## Sakila Database Questions

**Question 4: Write an SQL query to find the customer\_id who has the highest number of rentals.**

SQL Query:

```
SELECT customer_id, COUNT(*) AS total_rentals
FROM rental
GROUP BY customer_id
ORDER BY total_rentals DESC
LIMIT 1;
```

The screenshot shows a database interface with a query result for the 'rental' table. The query is: `SELECT customer_id, total_rentals FROM rental`. The result shows 148 rows and 46 columns. The 'customer\_id' column is highlighted in blue, and the 'total\_rentals' column is highlighted in yellow. The 'Cost' is 32ms.

customer_id	total_rentals
148	46

Explanation:

We count rentals per customer and select the maximum.

**Question 5: Write an SQL query to identify the month with the most rentals.**

SQL Query:

```
SELECT MONTH(rental_date) AS rental_month, COUNT(*) AS total_rentals
FROM rental
GROUP BY rental_month
ORDER BY total_rentals DESC
LIMIT 1;
```

The screenshot shows a database interface with a query result for the 'rental' table. The query is: `SELECT MONTH(rental_date) AS rental_month, COUNT(*) AS total_rentals FROM rental GROUP BY rental_month ORDER BY total_rentals DESC LIMIT 1;`. The result shows 7 rows and 6709 columns. The 'rental\_month' column is highlighted in blue, and the 'total\_rentals' column is highlighted in yellow. The 'Cost' is 15ms.

rental_month	total_rentals
7	6709

Explanation:

We extract the month from rental\_date and count rentals per month.

**Question 6: Find the total revenue generated per day.**

SQL Query:

```
SELECT DATE(payment_date) AS payment_day, SUM(amount) AS total_revenue
FROM payment
GROUP BY payment_day;
```

Q	Search Results	⚙️	✉️	👤	+	+	🗑️	↺	↻	⬆️	Export	▶️	👁️	Cost: 23ms
Q	payment_day	⬆️	🗑️	total_revenue	⬆️	🗑️								
>	2005-05-25			573.63										
>	2005-05-28			804.04										
>	2005-06-15			1376.52										
>	2005-06-16			1349.76										
>	2005-06-18			1486.56										
>	2005-06-21			1161.25										
>	2005-07-08			2210.88										

Explanation:

Payments are grouped by date to calculate daily revenue.

**Question 7: Find the store that generated the highest total revenue.**

SQL Query:

```
SELECT s.store_id, SUM(p.amount) AS total_revenue
FROM store s
JOIN staff st ON s.store_id = st.store_id
JOIN payment p ON st.staff_id = p.staff_id
GROUP BY s.store_id
ORDER BY total_revenue DESC
LIMIT 1;
```

Q	Search Results	⚙️	✉️	👤	+	+	🗑️	↺	↻	⬆️	Export	▶️	👁️	Cost: 20ms
store_id	total_revenue	⬆️	🗑️											
tinyint	decimal													
2	33924.06													

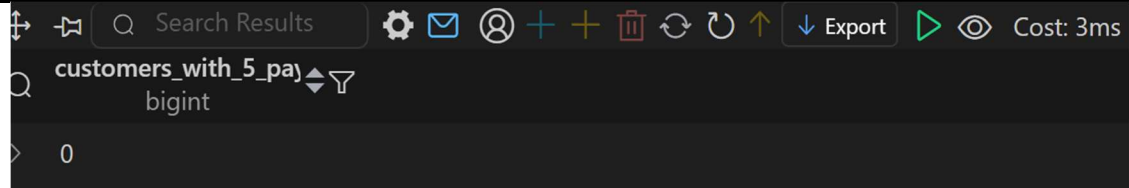
Explanation:

We join store, staff, and payment tables and sum revenue per store.

**Question 8: Find the customers who have made exactly 5 payments.**

SQL Query:

```
SELECT
  COUNT(*) AS customers_with_5_payments
FROM (
  SELECT customer_id
  FROM payment
  GROUP BY customer_id
  HAVING COUNT(*) = 5
) t;
```



The screenshot shows a SQL query execution interface. At the top, there is a search bar labeled "Search Results". Below it, a table is displayed with the column name "customers\_with\_5\_payments" and a data type of "bigint". The table contains a single row with the value "0". The interface also includes various icons for settings, email, and other functions, as well as an "Export" button and a "Cost: 3ms" indicator.

customers_with_5_payments
0

Explanation:

We use HAVING to filter customers with exactly five payments but there are no such customers who have made exactly 5 payments.

Shivansh