

SQL for Data Analysis

Objective: Use SQL queries to extract and analyze data from a database.

1. Select query is use to extract the data from database.

The screenshot shows a database interface with a query editor at the top containing the SQL command:

```
1 select * from play_store_data_2;
```

Below the query editor is a results grid. The grid has a header row with columns: App, Category, Rating, Reviews, Size, Installs, Type, Price, and Content R. The data rows show various photography apps:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content R
1	Font Studio- Photo Texts Image	PHOTOGRAPHY	4.2	197295	24M	10,000,000+	Free	0	Everyone
2	Add Text To Photo	PHOTOGRAPHY	4.1	21578	1.6M	1,000,000+	Free	0	Everyone
3	Phonto - Text on Photos	PHOTOGRAPHY	4.3	307453	17M	10,000,000+	Free	0	Everyone
4	Collage&Add Stickers papelook	PHOTOGRAPHY	4	32896	31M	5,000,000+	Free	0	Everyone
5	Shutterfly: Free Prints, Photo Books, Cards, Gifts	PHOTOGRAPHY	4.6	98717	59M	5,000,000+	Free	0	Everyone
6	Photo Collage - InstaMag	PHOTOGRAPHY	4.3	542561	46M	10,000,000+	Free	0	Everyone

2. Select query and where is use exract the data from database.

Where is used to specify conditions.

The screenshot shows a database interface with a query editor at the top containing the SQL command:

```
1 select * from play_store_data_2
2 where Rating>4;
```

Below the query editor is a results grid. The grid has a header row with columns: App, Category, Rating, Reviews, and Size. The data rows show productivity and parenting apps with a rating greater than 4:

	App	Category	Rating	Reviews	Size
463	Cloud Print	PRODUCTIVITY	4.1	282460	Var
464	Baby Names	PARENTING	4.5	86	2.8
465	Development of the child up to a year	PARENTING	4.6	1413	141
466	Favorite children's songs	PARENTING	4.7	39	951
467	Mozart Baby Sleep	PARENTING	4.7	107	981
468	Kids Videos	PARENTING	4.1	559	6.4

3. The ORDER BY clause in SQL is used to **sort the results of a query** based on one or more columns.

- By default, it sorts in **ascending order (ASC)**.
- You can explicitly specify **descending order (DESC)**.
- You can sort by **multiple columns** (first by one, then by another).

playstore_data

```
1 select * from play_store_data_2
2 order by Reviews Asc;
3
```

Grid view Form view

Total rows loaded: 2589

	App	Category	Rating	Reviews	Size	Installs	Type
1	Video Wallpaper Show	VIDEO_PLAYERS	NULL	0	13M	500+	Free
2	Norwegian For Kids & Babies F	FAMILY	NULL	0	14M	5+	Paid
3	F-1 watchface by Delta	PERSONALIZATION	NULL	0	Varies with device	10+	Paid
4	F Length Sim (no Ads)	PHOTOGRAPHY	NULL	0	1.7M	10+	Paid
5	G-NetReport Pro	TOOLS	NULL	0	1.6M	10+	Paid
6	D+H Reaction Wall	GAME	NULL	0	Varies with device	1+	Paid

playstore_data

```
1 select * from play_store_data_2
2 order by Reviews desc;
3
```

Grid view Form view

Total rows loaded: 2589

	App	Category	Rating	Reviews	Size	Installs	Type
1	Facebook	SOCIAL	4.1	78128208	Varies with device	1,000,000,000+	Free
2	WhatsApp Messenger	COMMUNICATION	4.4	69109672	Varies with device	1,000,000,000+	Free
3	Instagram	SOCIAL	4.5	66509917	Varies with device	1,000,000,000+	Free
4	Clash of Clans	FAMILY	4.6	44881447	98M	100,000,000+	Free
5	Clean Master- Space Cleaner & Antivirus	TOOLS	4.7	42916526	Varies with device	500,000,000+	Free
6	Subway Surfers	GAME	4.5	27711703	76M	1,000,000,000+	Free

4. The GROUP BY clause is used to **arrange identical data into groups**.

It's almost always paired with **aggregate functions** (SUM, AVG, COUNT, MIN, MAX).

Instead of returning every row, it summarizes data **per group**.

The screenshot shows a database interface with a toolbar at the top and a main area divided into two sections: a query editor and a results grid.

Query Editor:

```
playstore_data
1 select * from play_store_data_2
2 group by type;
3 |
```

Results Grid:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content R	Genres	Last Updated	C
1 Font Studio- Photo Texts Image	PHOTOGRAPHY	4.2	197295	24M	10,000,000+	Free	0	Everyone	Photography	2017-06-23 00:00:00	4.
2 Golfshot Plus: Golf GPS	SPORTS	4.1	3387	25M	50,000+	Paid	29.99	Everyone	Sports	2018-07-11 00:00:00	4.

The screenshot shows a database interface with a toolbar at the top and a main area divided into two sections: a query editor and a results grid.

Query Editor:

```
playstore_data
1 select * from play_store_data_2
2 group by Category$;
3 |
```

Results Grid:

App	Category	Rating	Reviews	Size	Installs	Type	Price	Content R	Genres	Last Updated	C
1 Cardi B Wallpaper	ART_AND DESIGN	4.8	253	3.7M	50,000+	Free	0	Everyone	Art & Design	2018-07-11 00:00:00	4.
2 MHD F-Series	AUTO_AND_VEHICLES	4.9	73	23M	1,000+	Free	0	Everyone	Automotive	2018-07-11 00:00:00	4.
3 AI Face Beauty Analysis - IntelliFace (Free)	BEAUTY	3.1	40	13M	10,000+	Free	0	Everyone	Beauty	2018-07-11 00:00:00	4.
4 Bible	BOOKS_AND_REFERENCE	4.7	2440695	Varies with device	100,000,000+	Free	0	Everyone	Books & Reference	2018-07-11 00:00:00	4.
5 R+F PULSE	BUSINESS	3.4	58	3.0M	10,000+	Free	0	Everyone	Business	2018-07-11 00:00:00	4.
6 Q Avatar (Avatar Maker)	COMICS	4.4	2012	3.6M	100,000+	Free	0	Everyone	Comics	2018-07-11 00:00:00	4.

5. An **INNER JOIN** returns rows that have **matching values in both tables**.

If there's no match, the row is excluded.

It's the most commonly used join for analysis.

The screenshot shows a database interface with a toolbar at the top and a query editor below it. The query is:

```
1 select * from play_store_data_2 p inner join user_reviews_data_2 u
2 on u.app=p.app;
3
```

The results are displayed in a grid view. The columns include App:1, Translated_Review, and several other app details. The grid shows 244 total rows loaded. The data includes reviews for the app "Cricbuzz - Live Cricket Scores & News".

	Genres	Last Updated	Current Ver	Android Ver	App:1	Translated_Review
1	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	, Bethe bethe worm will be cut, you will always hav
2	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	Ads ok! But too many ads. I really hate it. Uninstall
3	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	All things dramatically change.one tha best cricket
4	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	Best cricket fans. The UI simple yet amazing, faster
5	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	work. One problem I faced news articles sometimes
6	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	Best keeping track matches, news latest videos. An
7	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	Edit: full intrusive ads. Cricbuzz leading content pl
8	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	rating 2/5. Original review: The thing ESPN Cricket
9	Sports	2018-05-18 00:00:00	Varies with device	Varies with device	Cricbuzz - Live Cricket Scores & News	Every vulgar app. makes difficult use... either office

6. A **LEFT JOIN** returns **all rows from the left table** and the **matching rows from the right table**.

If there's no match in the right table, the result will still include the left table row, but with **NULLS** for the right table's columns.

It's useful when you want to see everything from one table, even if related data doesn't exist in the other.

The screenshot shows a database interface with a toolbar at the top and a query editor below it. The query is:

```
1 select * from play_store_data_2 p left join user_reviews_data_2 u
2 on u.app=p.app;
3
```

The results are displayed in a grid view. The columns include App, Category, Rating, Reviews, and Size. The grid shows 2828 total rows loaded. It includes rows for various photo editing apps like "Font Studio- Photo Texts Image" and "Shutterfly: Free Prints, Photo Books, Cards, Gifts".

	App	Category	Rating	Reviews	Size
1	Font Studio- Photo Texts Image	PHOTOGRAPHY	4.2	197295	24M
2	Add Text To Photo	PHOTOGRAPHY	4.1	21578	1.6M
3	Phonto - Text on Photos	PHOTOGRAPHY	4.3	307453	17M
4	Collage&Add Stickers papelook	PHOTOGRAPHY	4	32896	31M
5	Shutterfly: Free Prints, Photo Books, Cards, Gifts	PHOTOGRAPHY	4.6	98717	59M
6	Photo Collage - InstaMag	PHOTOGRAPHY	4.3	542561	46M
7	Meitu - Beauty Cam, Easy Photo Editor	PHOTOGRAPHY	4.5	462702	45M
8	ESPN	SPORTS	4.2	521138	Varies with devi
9	Free Sports TV	SPORTS	4.3	1802	9.8M

7. A **RIGHT JOIN** returns **all rows from the right table** and the **matching rows from the left table**.

If there's no match in the left table, the result will still include the right table row, but with **NULLs** for the left table's columns.

It's basically the opposite of a **LEFT JOIN**.

The screenshot shows a database interface with a toolbar at the top and a query editor below it. The query editor contains the following SQL code:

```
1 select * from play_store_data_2 p right join user_reviews_data_2 u
2 on u.app=p.app;
3
```

Below the query editor, there are two tabs: "Grid view" and "Form view". The "Grid view" tab is selected, and the results are displayed in a table. The table has the following columns: App, First page, Category, Rating, Reviews, Size, Installs, Type, Price, Content Rating, and Genres. The data shows 930 rows for the app "Cricbuzz - Live Cricket Scores & News", which is listed 1 through 9. All rows have a rating of 4.5, reviews count of 838765, size as "Varies with device", installs count of 50,000,000+, type as "Free", price as 0, content rating as "Everyone", and genres as "Sports".

	App	First page	Category	Rating	Reviews	Size	Installs	Type	Price	Content R	Genres
1	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
2	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
3	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
4	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
5	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
6	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
7	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
8	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
9	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports
	Cricbuzz - Live Cricket Scores & News		SPORTS	4.5	838765	Varies with device	50,000,000+	Free	0	Everyone	Sports

8. A **subquery** is a query **inside another query**.

It's enclosed in parentheses () .

It can return a **single value**, a **list of values**, or even a **table**.

Subqueries are often used in WHERE, FROM, or SELECT clauses.

The screenshot shows a database interface with a toolbar at the top. The toolbar includes icons for saving, opening, printing, and various file operations. Below the toolbar, there are two tabs: "Query" and "History". The "Query" tab is selected, displaying the following SQL code:

```
1 select max(reviews) from play_store_data_2
2 where reviews < (select max(reviews) from play_store_data_2)
```

Below the code, the results are displayed in a grid view. The results show one row with the column header "max(reviews)" and the value "69109672". The interface also includes navigation buttons (back, forward, search, etc.) and a message "Total rows loaded: 1".

The screenshot shows a database interface with a toolbar at the top. The toolbar includes icons for saving, opening, printing, and various file operations. Below the toolbar, there are two tabs: "Query" and "History". The "Query" tab is selected, displaying the following SQL code:

```
1 select app, category, rating from play_store_data_2
2 WHERE size > (
3     SELECT AVG(size) FROM play_store_data_2
4 );
```

Below the code, the results are displayed in a grid view. The results show 1834 rows, each containing "App", "Category", and "Rating" columns. The first few rows are:

App	Category	Rating
Collage&Add Stickers papelook	PHOTOGRAPHY	4
Shutterfly: Free Prints, Photo Books, Cards, Gifts	PHOTOGRAPHY	4.6
Photo Collage - InstaMag	PHOTOGRAPHY	4.3
Meitu – Beauty Cam, Easy Photo Editor	PHOTOGRAPHY	4.5
ESPN	SPORTS	4.2
Free Sports TV	SPORTS	4.3
LiveScore: Live Sport Updates	SPORTS	4.4
MLB At Bat	SPORTS	4.2
NFL	SPORTS	4.1
theScore: Live Sports Scores, News, Stats & Videos	SPORTS	4.4

The interface also includes navigation buttons (back, forward, search, etc.) and a message "Total rows loaded: 1834".

playstore_data

Query History

```
1 select app, category, rating from play_store_data_2
2 WHERE rating > (
3     SELECT AVG(rating) FROM play_store_data_2
4 );
5
```

Grid view Form view

Total rows loaded: 1583

	App	Category	Rating
1	Font Studio- Photo Texts Image	PHOTOGRAPHY	4.2
2	Phonto - Text on Photos	PHOTOGRAPHY	4.3
3	Shutterfly: Free Prints, Photo Books, Cards, Gifts	PHOTOGRAPHY	4.6
4	Photo Collage - InstaMag	PHOTOGRAPHY	4.3
5	Meitu – Beauty Cam, Easy Photo Editor	PHOTOGRAPHY	4.5
6	ESPN	SPORTS	4.2
7	Free Sports TV	SPORTS	4.3
8	LiveScore: Live Sport Updates	SPORTS	4.4
9	MLB At Bat	SPORTS	4.2
10	theScore: Live Sports Scores, News, Stats & Videos	SPORTS	4.4

9. COUNT() returns the number of rows that match a condition.

It can count **all rows** (COUNT(*)) or **non-NULL values** in a specific column(COUNT(column_name)).

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
1 select count(*) from play_store_data_2
2 where rating>3;
```

The results pane shows a single row of data:

	count(*)
1	2330

Below the results, it says "Total rows loaded: 1".

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL code:

```
1 select count(*) from play_store_data_2;
```

The results pane shows a single row of data:

	count(*)
1	2589

Below the results, it says "Total rows loaded: 1".

10. Sum() adds up the values in a numeric column.

Often used with GROUP BY to calculate totals per category, customer, or product.

Works only on numeric data types (e.g., integers, decimals).

The screenshot shows a database interface with a toolbar at the top, a query editor, and a results grid. The toolbar includes icons for file operations, search, and export. The query editor contains the SQL command: `1 select sum(installs) from play_store_data_2;`. The results grid shows one row with the sum of installs: `sum(installs)` and `172499`.

	sum(installs)
1	172499

Total rows loaded: 1

The screenshot shows a database interface with a toolbar at the top, a query editor, and a results grid. The toolbar includes icons for file operations, search, and export. The query editor contains the SQL command: `1 select sum(reviews) from play_store_data_2;`. The results grid shows one row with the sum of reviews: `sum(reviews)` and `1507911152`.

	sum(reviews)
1	1507911152

Total rows loaded: 1

11. `AVG()` calculates the **average value** of a numeric column.

Often used with `GROUP BY` to find averages per category, customer, or product.
Ignores `NONE` values automatically.

The screenshot shows a database interface with a toolbar at the top. The toolbar includes icons for database connection, export, text editor, table, search, and file operations. Below the toolbar, there are tabs for "Query" and "History", with "Query" selected. A code editor window contains the SQL query: `1 select avg(price) from play_store_data_2;`. The results pane below shows the output of the query: "avg(price)" and "2.5842448821939". The results are displayed in a grid view. At the bottom of the results pane, it says "Total rows loaded: 1".

	avg(price)
1	2.5842448821939

The screenshot shows a database interface with a toolbar at the top. The toolbar includes icons for database connection, export, text editor, table, search, and file operations. Below the toolbar, there are tabs for "Query" and "History", with "Query" selected. A code editor window contains the SQL query: `1 select avg(size) from play_store_data_2;`. The results pane below shows the output of the query: "avg(size)" and "28.29354963306296". The results are displayed in a grid view. At the bottom of the results pane, it says "Total rows loaded: 1".

	avg(size)
1	28.29354963306296

12. MIN() returns the **smallest value** in a column.

Often used to find the **earliest date, lowest price, smallest size, or minimum rating**.

Ignores NULL values automatically.

The screenshot shows a database interface with a toolbar at the top. The toolbar includes icons for saving, printing, and various database operations. Below the toolbar is a menu bar with 'Query' and 'History' tabs, where 'Query' is selected. The main area contains a SQL query:

```
1 select min(price) from play_store_data_2;
```

Below the query results, there are two tabs: 'Grid view' and 'Form view'. The 'Grid view' tab is selected. A message 'Total rows loaded: 1' is displayed. The result is shown in a grid:

	min(price)
1	0

The screenshot shows a database interface with a toolbar at the top. The toolbar includes icons for saving, printing, and various database operations. Below the toolbar is a menu bar with 'Query' and 'History' tabs, where 'Query' is selected. The main area contains a SQL query:

```
1 select app,min(size) from play_store_data_2;
2 |
```

Below the query results, there are two tabs: 'Grid view' and 'Form view'. The 'Grid view' tab is selected. A message 'Total rows loaded: 1' is displayed. The result is shown in a grid:

	App	min(size)
1	OnePlus Icon Pack - Square	1.1M

13. MAX() returns the **largest value** in a column.

Often used to find the **highest rating, most installs, latest date, or maximum price**.

Ignores NULL values automatically.

The screenshot shows a database interface with a toolbar at the top. Below the toolbar, there are tabs for "Query" and "History", with "Query" selected. A code editor window contains the following SQL query:

```
1 select max(reviews) from play_store_data_2
```

Below the code editor, there are two viewing options: "Grid view" and "Form view", with "Grid view" selected. At the bottom of the interface, there is a toolbar with various icons and a status bar displaying "Total rows loaded: 1". The main results area displays the following data:

	max(reviews)
1	78128208

The screenshot shows a database interface with a toolbar at the top. Below the toolbar, there are tabs for "Query" and "History", with "Query" selected. A code editor window contains the following SQL query:

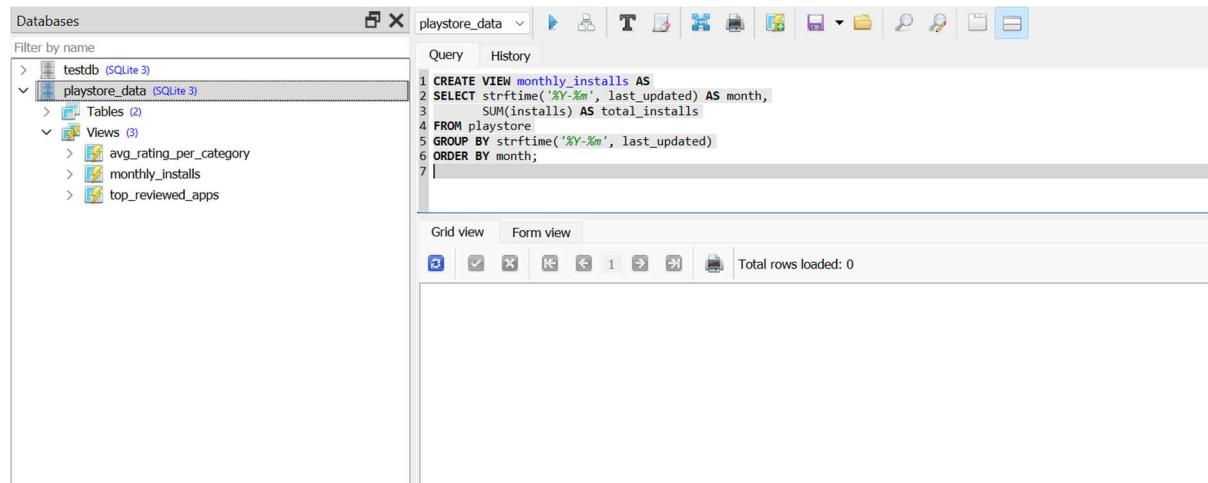
```
1 select max(price) from play_store_data_2;
```

Below the code editor, there are two viewing options: "Grid view" and "Form view", with "Grid view" selected. At the bottom of the interface, there is a toolbar with various icons and a status bar displaying "Total rows loaded: 1". The main results area displays the following data:

	max(price)
1	400

14. SQL views for analysis. Views are like reusable “virtual tables” that store complex queries so you can query them directly for insights.

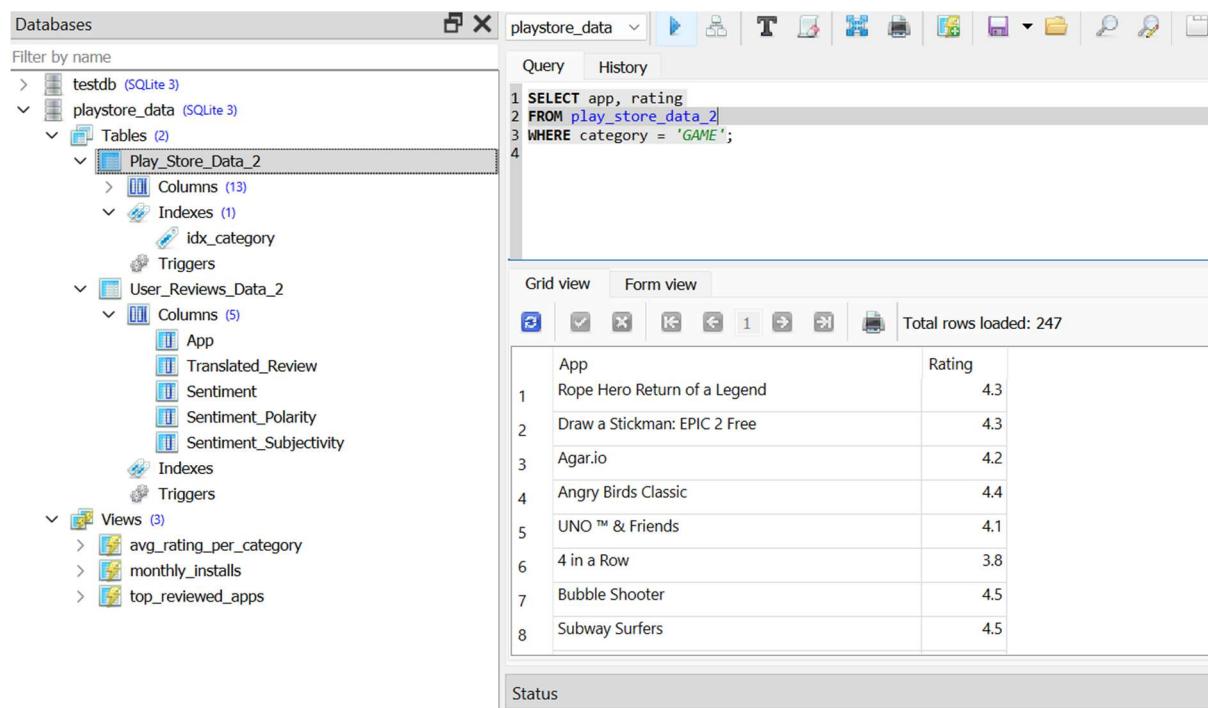
They’re especially useful in analytics because they simplify repeated queries and make dashboards easier to build.



The screenshot shows a database management interface with the following details:

- Databases:** A tree view on the left shows two databases: "testdb (SQLite 3)" and "playstore_data (SQLite 3)".
- Query:** The main area displays a SQL query for creating a view named "monthly_installs".

```
1 CREATE VIEW monthly_installs AS
2 SELECT strftime('%Y-%m', last_updated) AS month,
3       SUM(installs) AS total_installs
4 FROM playstore
5 GROUP BY strftime('%Y-%m', last_updated)
6 ORDER BY month;
```

The screenshot shows a database management interface with the following details:

- Databases:** A tree view on the left shows three databases: "testdb (SQLite 3)", "playstore_data (SQLite 3)", and "User_Reviews_Data_2".
- Query:** The main area displays a SQL query for selecting app and rating from the "play_store_data_2" table where the category is 'GAME'.

```
1 SELECT app, rating
2 FROM play_store_data_2
3 WHERE category = 'GAME';
4
```

The results grid shows the following data:

	App	Rating
1	Rope Hero Return of a Legend	4.3
2	Draw a Stickman: EPIC 2 Free	4.3
3	Agar.io	4.2
4	Angry Birds Classic	4.4
5	UNO ™ & Friends	4.1
6	4 in a Row	3.8
7	Bubble Shooter	4.5
8	Subway Surfers	4.5