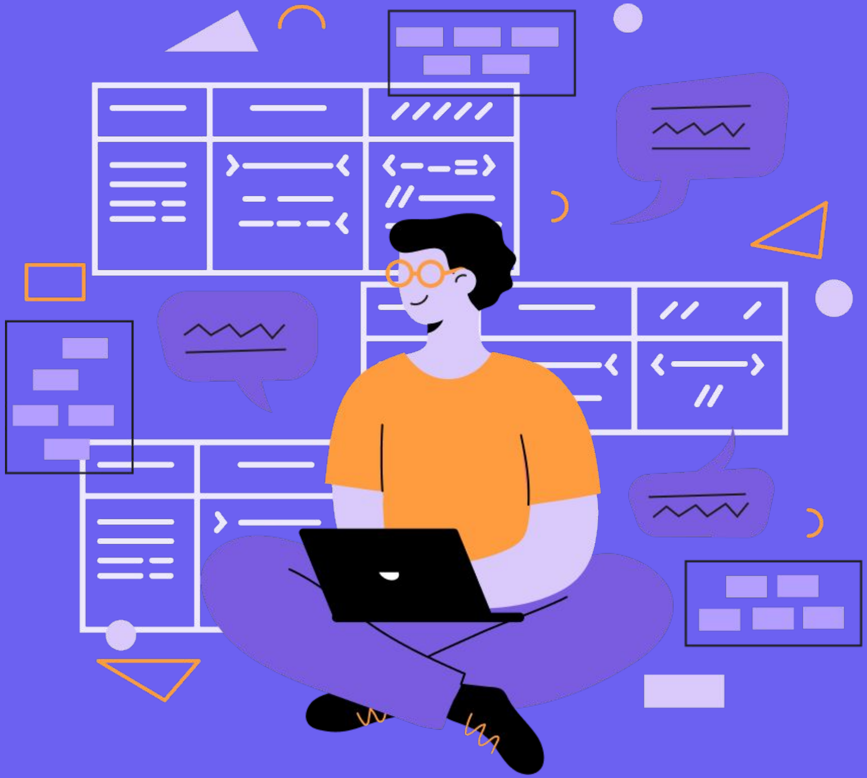
Subqueries



Relevel
by Unacademy

Subqueries

What is a sub-query?

A SQL subquery is a query that is contained within another query. They are used to execute a query dependent on the outcome of another query. Subqueries allow us to accomplish this without writing two distinct queries and copy-pasting the results.

Key Characteristic of a sub-query:

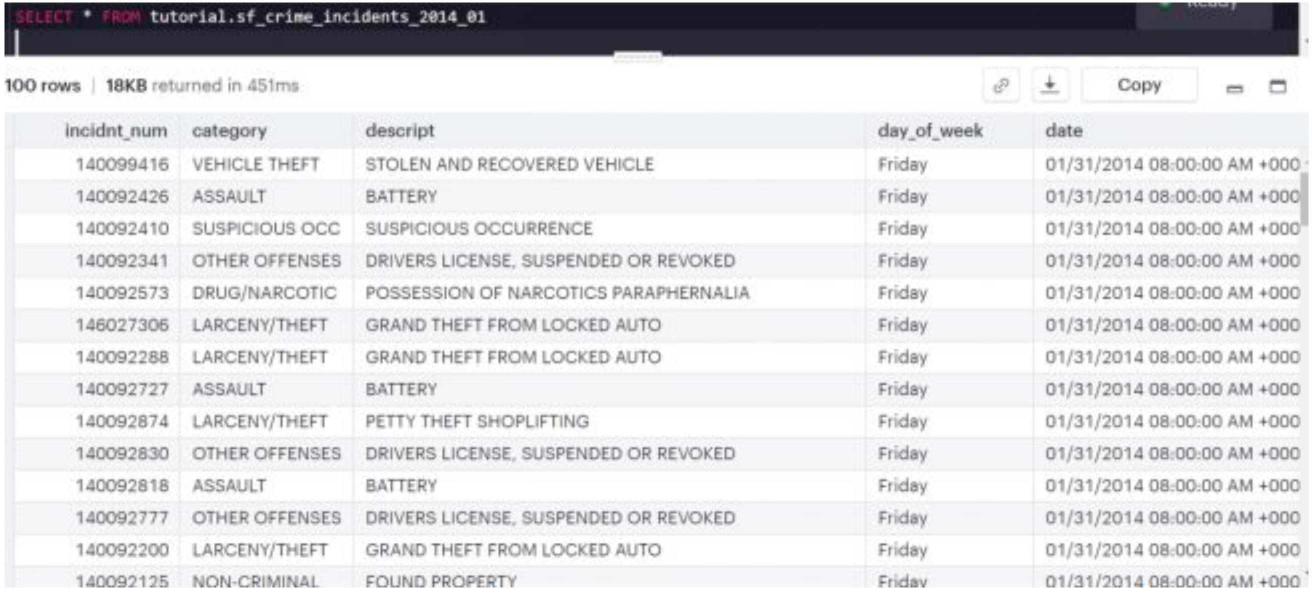
- Sub-queries should be provided with an alias
- WHERE or HAVING clauses contain subqueries
- Individual values or a list of records can be returned via subqueries
- Sub Questions must be surrounded by brackets ()

There isn't such a thing as generic syntax. Subqueries are normal queries that are enclosed in parenthesis. Subqueries can be utilised in various ways and at various points within a query. We will learn subqueries through various use cases.

Subqueries

Understanding subqueries with an example:

We will use tutorial.sf_crime_incidents_2014_01 of mode.com to understand this concept.



```
SELECT * FROM tutorial.sf_crime_incidents_2014_01
```

100 rows | 18KB returned in 451ms

incidnt_num	category	descript	day_of_week	date
140099416	VEHICLE THEFT	STOLEN AND RECOVERED VEHICLE	Friday	01/31/2014 08:00:00 AM +000
140092426	ASSAULT	BATTERY	Friday	01/31/2014 08:00:00 AM +000
140092410	SUSPICIOUS OCC	SUSPICIOUS OCCURRENCE	Friday	01/31/2014 08:00:00 AM +000
140092341	OTHER OFFENSES	DRIVERS LICENSE, SUSPENDED OR REVOKED	Friday	01/31/2014 08:00:00 AM +000
140092573	DRUG/NARCOTIC	POSSESSION OF NARCOTICS PARAPHERNALIA	Friday	01/31/2014 08:00:00 AM +000
146027306	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	01/31/2014 08:00:00 AM +000
140092288	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	01/31/2014 08:00:00 AM +000
140092727	ASSAULT	BATTERY	Friday	01/31/2014 08:00:00 AM +000
140092874	LARCENY/THEFT	PETTY THEFT SHOPLIFTING	Friday	01/31/2014 08:00:00 AM +000
140092830	OTHER OFFENSES	DRIVERS LICENSE, SUSPENDED OR REVOKED	Friday	01/31/2014 08:00:00 AM +000
140092818	ASSAULT	BATTERY	Friday	01/31/2014 08:00:00 AM +000
140092777	OTHER OFFENSES	DRIVERS LICENSE, SUSPENDED OR REVOKED	Friday	01/31/2014 08:00:00 AM +000
140092200	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	01/31/2014 08:00:00 AM +000
140092125	NON-CRIMINAL	FOUND PROPERTY	Friday	01/31/2014 08:00:00 AM +000

Basic subquery

Find all the crimes which happened in SF on a Friday and are still unresolved.

Query

```
SELECT sub.*
  FROM (
        SELECT *
          FROM tutorial.sf_crime_incidents_2014_01
         WHERE day_of_week = 'Friday'
       ) sub
 WHERE sub.resolution = 'NONE'
```

Basic subquery

Let’s break down this query and understand:

The database first executes the "inner query"—the part between the parentheses:

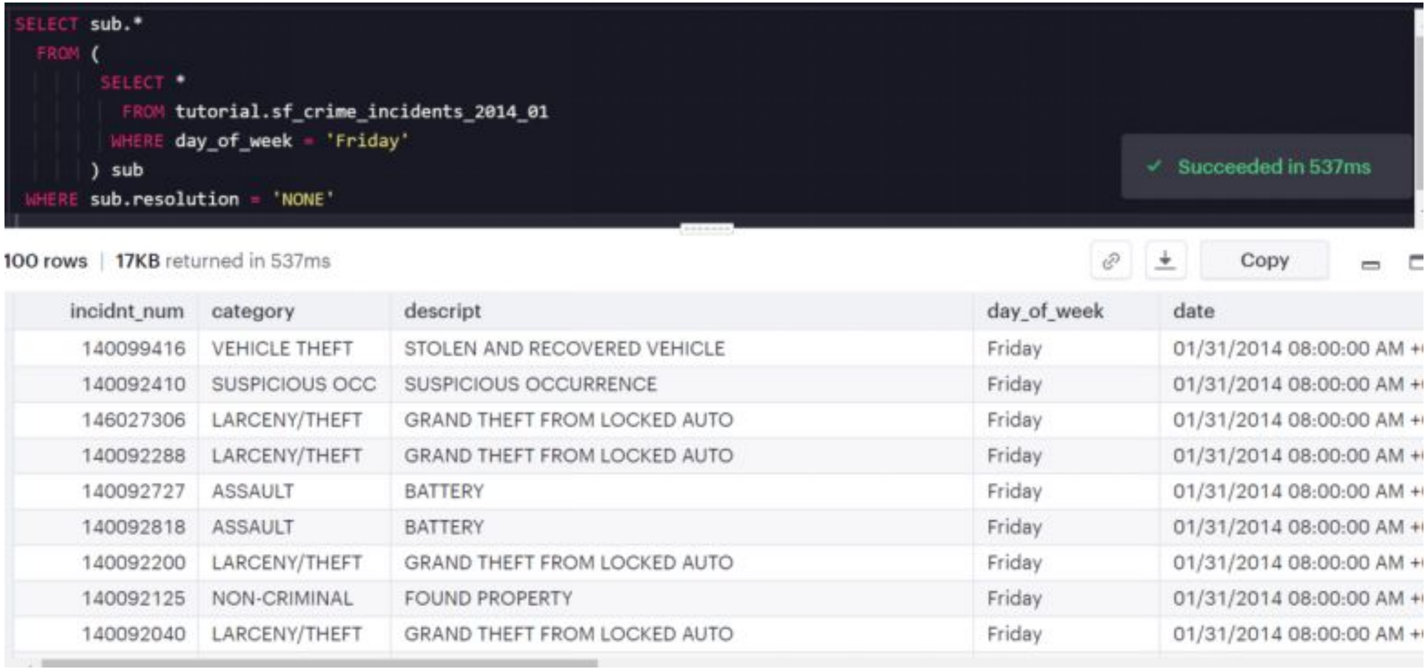
Query

```
SELECT *
  FROM tutorial.sf_crime_incidents_2014_01
 WHERE day_of_week = 'Friday'
```

It will give us a list of all the crimes happening on a Friday. The output of the sub-query will be consumed and will give us filtered data where cases are still unresolved.

Basic subquery

Output



```
SELECT sub.*
  FROM (
        SELECT *
          FROM tutorial.sf_crime_incidents_2014_01
         WHERE day_of_week = 'Friday'
       ) sub
 WHERE sub.resolution = 'NONE'
```

✓ Succeeded in 537ms

100 rows | 17KB returned in 537ms

Copy

incidnt_num	category	descript	day_of_week	date
140099416	VEHICLE THEFT	STOLEN AND RECOVERED VEHICLE	Friday	01/31/2014 08:00:00 AM +
140092410	SUSPICIOUS OCC	SUSPICIOUS OCCURRENCE	Friday	01/31/2014 08:00:00 AM +
146027306	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	01/31/2014 08:00:00 AM +
140092288	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	01/31/2014 08:00:00 AM +
140092727	ASSAULT	BATTERY	Friday	01/31/2014 08:00:00 AM +
140092818	ASSAULT	BATTERY	Friday	01/31/2014 08:00:00 AM +
140092200	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	01/31/2014 08:00:00 AM +
140092125	NON-CRIMINAL	FOUND PROPERTY	Friday	01/31/2014 08:00:00 AM +
140092040	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Friday	01/31/2014 08:00:00 AM +

Subquery With Aggregate Functions

Find the average number of crimes across the different days of the week per month:

Query

```
SELECT LEFT(sub.date, 2) AS cleaned_month,
       sub.day_of_week,
       AVG(sub.incidents) AS average_incidents
 FROM (
        SELECT day_of_week,
               date,
               COUNT(incidnt_num) AS incidents
          FROM tutorial.sf_crime_incidents_2014_01
         GROUP BY 1,2
       ) sub
 GROUP BY 1,2
 ORDER BY 1,2
```

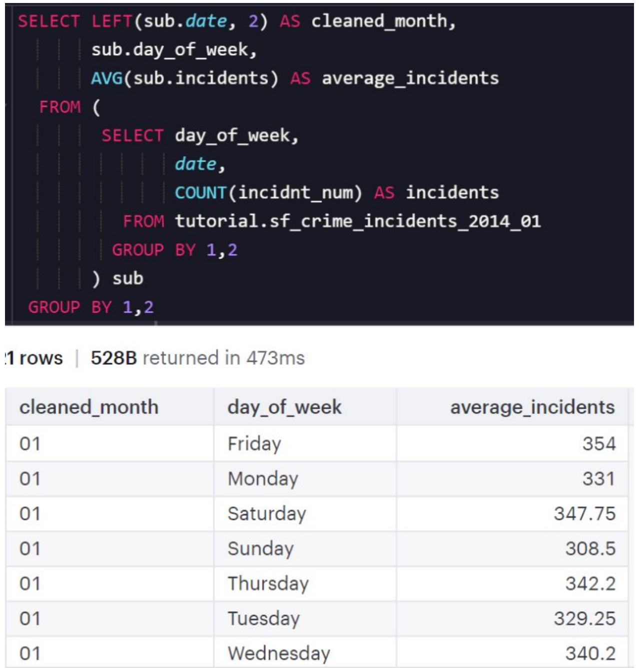
Subquery With Aggregate Functions

Let’s break down this query and understand:

- The inner query counts the number of incidents happening on a particular date and find the day of the week associated with that date
- The outer query finds the month associated with the date and give us the average incidents per month on a day of the week.:

Subquery With Aggregate Functions

Output:



```
SELECT LEFT(sub.date, 2) AS cleaned_month,
       sub.day_of_week,
       AVG(sub.incidents) AS average_incidents
  FROM (
        SELECT day_of_week,
               date,
               COUNT(incidnt_num) AS incidents
          FROM tutorial.sf_crime_incidents_2014_01
         GROUP BY 1,2
       ) sub
 GROUP BY 1,2
```

1 rows | 528B returned in 473ms

cleaned_month	day_of_week	average_incidents
01	Friday	354
01	Monday	331
01	Saturday	347.75
01	Sunday	308.5
01	Thursday	342.2
01	Tuesday	329.25
01	Wednesday	340.2

Subqueries with conditional logic

Find all the data in the dataset for the first time a crime is recorded in the dataset.

Query

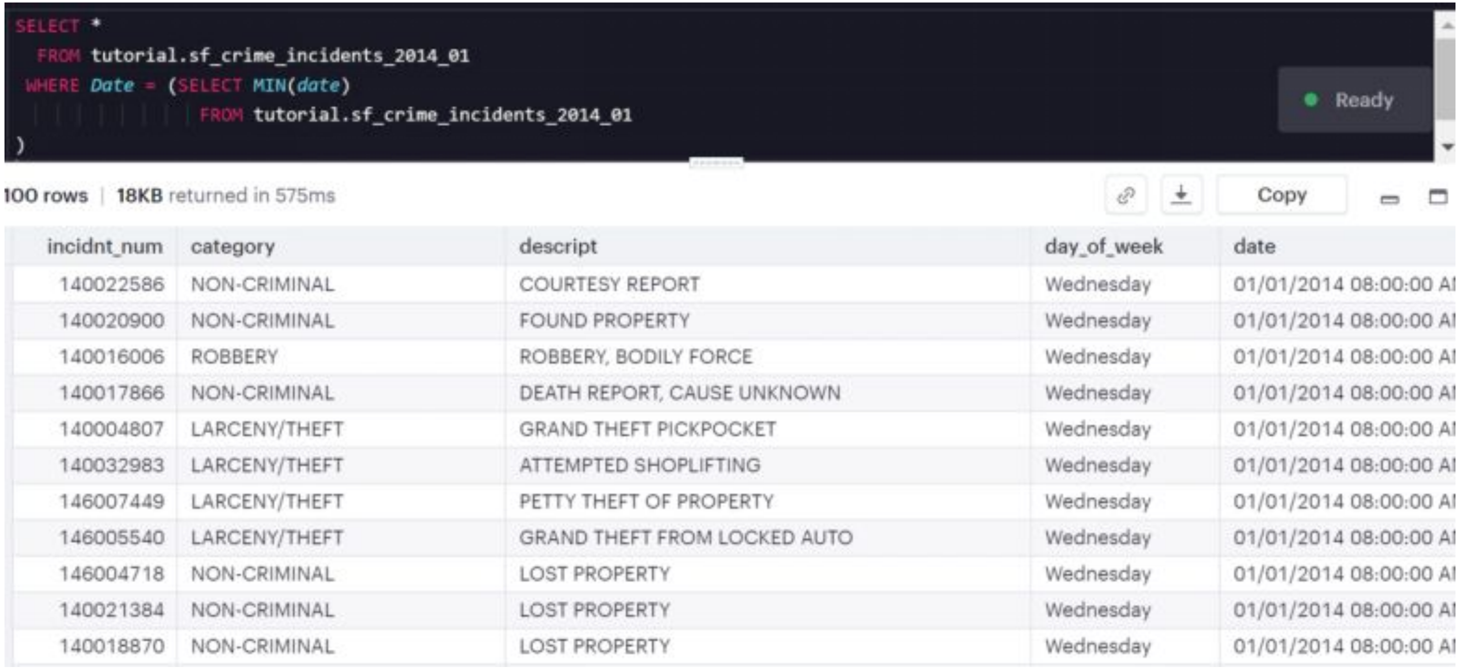
```
SELECT *
  FROM tutorial.sf_crime_incidents_2014_01
 WHERE Date = (SELECT MIN(date)
                 FROM tutorial.sf_crime_incidents_2014_01
              )
```

Let's break down this query and understand:

- The inner query finds the minimum date for the dataset
- The outer query gives all the records against the minimum date.

Subqueries with conditional logic

Output:



```
SELECT *
  FROM tutorial.sf_crime_incidents_2014_01
 WHERE Date = (SELECT MIN(date)
                FROM tutorial.sf_crime_incidents_2014_01
)
```

Ready

100 rows | 18KB returned in 575ms

Copy

incidnt_num	category	descript	day_of_week	date
140022586	NON-CRIMINAL	COURTESY REPORT	Wednesday	01/01/2014 08:00:00 A
140020900	NON-CRIMINAL	FOUND PROPERTY	Wednesday	01/01/2014 08:00:00 A
140016006	ROBBERY	ROBBERY, BODILY FORCE	Wednesday	01/01/2014 08:00:00 A
140017866	NON-CRIMINAL	DEATH REPORT, CAUSE UNKNOWN	Wednesday	01/01/2014 08:00:00 A
140004807	LARCENY/THEFT	GRAND THEFT PICKPOCKET	Wednesday	01/01/2014 08:00:00 A
140032983	LARCENY/THEFT	ATTEMPTED SHOPLIFTING	Wednesday	01/01/2014 08:00:00 A
146007449	LARCENY/THEFT	PETTY THEFT OF PROPERTY	Wednesday	01/01/2014 08:00:00 A
146005540	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	01/01/2014 08:00:00 A
146004718	NON-CRIMINAL	LOST PROPERTY	Wednesday	01/01/2014 08:00:00 A
140021384	NON-CRIMINAL	LOST PROPERTY	Wednesday	01/01/2014 08:00:00 A
140018870	NON-CRIMINAL	LOST PROPERTY	Wednesday	01/01/2014 08:00:00 A

Subqueries with Join

Add a column in the dataset which displays the total number of incidents associated with a date.

Query

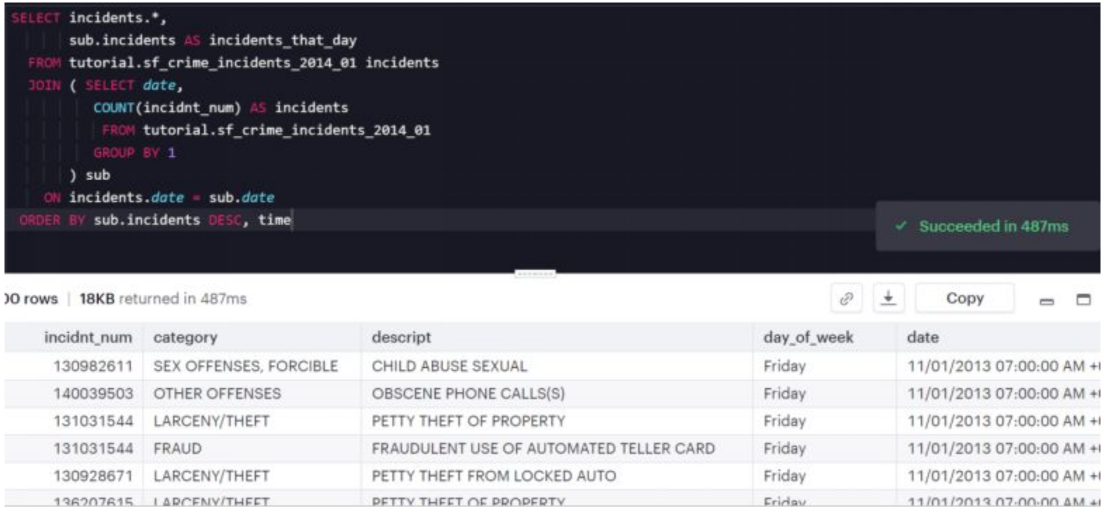
```
SELECT incidents.*,
       sub.incidents AS incidents_that_day
  FROM tutorial.sf_crime_incidents_2014_01 incidents
  JOIN ( SELECT date,
  COUNT(incidnt_num) AS incidents
  FROM tutorial.sf_crime_incidents_2014_01
  GROUP BY 1
  ) sub
 ON incidents.date = sub.date
ORDER BY sub.incidents DESC, time
```

Subqueries with Join

Let’s break down this query and understand:

- The inner query finds the count of all the incidents that happened in a day
- The outer query joins the count with the dataset

Output:



```
SELECT incidents.*,
       sub.incidents AS incidents_that_day
  FROM tutorial.sf_crime_incidents_2014_01 incidents
  JOIN ( SELECT date,
          COUNT(incidnt_num) AS incidents
           FROM tutorial.sf_crime_incidents_2014_01
          GROUP BY 1
       ) sub
    ON incidents.date = sub.date
 ORDER BY sub.incidents DESC, time
```

✓ Succeeded in 487ms

0 rows | 18KB returned in 487ms

Copy

incidnt_num	category	descript	day_of_week	date
130982611	SEX OFFENSES, FORCIBLE	CHILD ABUSE SEXUAL	Friday	11/01/2013 07:00:00 AM +
140039503	OTHER OFFENSES	OBSCENE PHONE CALLS(S)	Friday	11/01/2013 07:00:00 AM +
131031544	LARCENY/THEFT	PETTY THEFT OF PROPERTY	Friday	11/01/2013 07:00:00 AM +
131031544	FRAUD	FRAUDULENT USE OF AUTOMATED TELLER CARD	Friday	11/01/2013 07:00:00 AM +
130928671	LARCENY/THEFT	PETTY THEFT FROM LOCKED AUTO	Friday	11/01/2013 07:00:00 AM +
136207615	LARCENY/THEFT	PETTY THEFT OF PROPERTY	Friday	11/01/2013 07:00:00 AM +

Types of subqueries

Two types of subqueries:

1)Nested query

2)Correlated Query

1. **Nested Subqueries:** A query is written inside another query in a nested query, and the result of the inner query is used in the execution of the outer query.
   The inner query is executed first and only once in Nested Query. The result of the Inner query is used to run the Outer query. As a result, the Inner query is employed in the execution of the Outer query.

Types of subqueries

Understanding Nested Query:

Assume we have two tables:

Orders:

Column	Data Type
OrderID	int
CustomerID	int
OrderDate	Date

Customers:

Column	Data Type
CustomerID	int
CustomerName	VARCHAR
ContactName	VARCHAR
Country	VARCHAR

Types of subqueries

Query

SELECT * FROM Customers WHERE

CustomerID IN (SELECT CustomerID FROM Orders);

Let's break down this query and understand:

- The inner query "SELECT CustomerID FROM Orders" will first run and give all the customer IDs.
- The outer query will then run and look for the customer id given by the inner query and display records for all such customers.

Types of subqueries

2. Correlated Query: There are also SQL subqueries where the inner query relies on information obtained from the outer query. The outcome of a subquery depends on the value of a column of its parent query table.

Understanding Correlated Query:

Assume we have two tables:

Orders:

Column	Data Type
OrderID	int
CustomerID	int
OrderDate	Date

Customers:

Column	Data Type
CustomerID	int
CustomerName	VARCHAR
ContactName	VARCHAR
Country	VARCHAR

Types of subqueries

Query

```
SELECT * FROM Customers where
EXISTS (SELECT CustomerID FROM Orders
WHERE Orders.CustomerID=Customers.CustomerID);
```

Let's break down this query and understand:

- Here firstly, the outer query is executed and lists all the customers.
- The inner query will then run and look for the customer id from the outer query and then see where the record matches with the customer id present in orders.

Practice Questions

Instructions:

- **We will use mode.com for all the practice questions.**
- **We will use following datasets in the questions below:**
  - tutorial.sf_crime_incidents_2014_01
  - tutorial.sf_crime_incidents_cleandate
  - tutorial.crunchbase_companies
  - tutorial.crunchbase_acquisitions
  - tutorial.crunchbase_investments_part1
  - tutorial.crunchbase_investments_part2

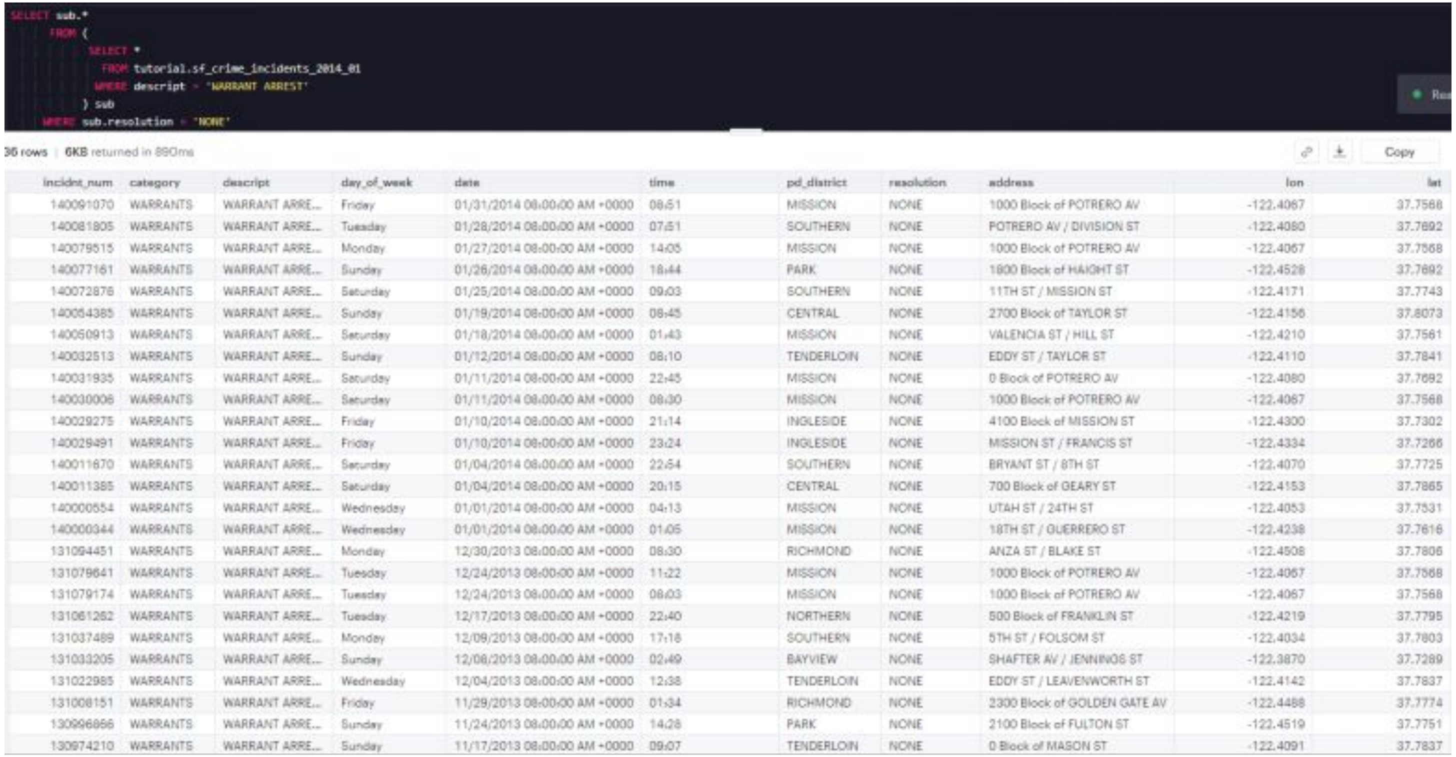
Practice Questions

Question-1: Create a query that selects all Warrant Arrests from the tutorial.sf_crime_incidents_2014_01 dataset, then wrap it in a query that only exposes unresolved incidents.

Answer-1:

```
SELECT sub.*
FROM (
SELECT *
FROM tutorial.sf_crime_incidents_2014_01
WHERE descript = 'WARRANT ARREST'
) sub
WHERE sub.resolution = 'NONE'
```

Practice Questions 1



```sql
SELECT sub.*
      FROM (
            SELECT *
              FROM tutorial.sf_crime_incidents_2014_01
             WHERE descript = 'WARRANT ARREST'
           ) sub
     WHERE sub.resolution = 'NONE'
```

36 rows | 6KB returned in 890ms

Copy

incidnt_num	category	descript	day_of_week	date	time	pd_district	resolution	address	lon	lat
140091070	WARRANTS	WARRANT ARRE...	Friday	01/31/2014 08:00:00 AM +0000	08:51	MISSION	NONE	1000 Block of POTRERO AV	-122.4067	37.7568
140081805	WARRANTS	WARRANT ARRE...	Tuesday	01/28/2014 08:00:00 AM +0000	07:51	SOUTHERN	NONE	POTRERO AV / DIVISION ST	-122.4080	37.7692
140079515	WARRANTS	WARRANT ARRE...	Monday	01/27/2014 08:00:00 AM +0000	14:05	MISSION	NONE	1000 Block of POTRERO AV	-122.4067	37.7568
140077161	WARRANTS	WARRANT ARRE...	Sunday	01/26/2014 08:00:00 AM +0000	18:44	PARK	NONE	1800 Block of HAIGHT ST	-122.4528	37.7692
140072876	WARRANTS	WARRANT ARRE...	Saturday	01/25/2014 08:00:00 AM +0000	09:03	SOUTHERN	NONE	11TH ST / MISSION ST	-122.4171	37.7743
140064385	WARRANTS	WARRANT ARRE...	Sunday	01/19/2014 08:00:00 AM +0000	08:45	CENTRAL	NONE	2700 Block of TAYLOR ST	-122.4156	37.8073
140060913	WARRANTS	WARRANT ARRE...	Saturday	01/18/2014 08:00:00 AM +0000	01:43	MISSION	NONE	VALENCIA ST / HILL ST	-122.4210	37.7561
140032513	WARRANTS	WARRANT ARRE...	Sunday	01/12/2014 08:00:00 AM +0000	08:10	TENDERLOIN	NONE	EDDY ST / TAYLOR ST	-122.4110	37.7841
140031935	WARRANTS	WARRANT ARRE...	Saturday	01/11/2014 08:00:00 AM +0000	22:45	MISSION	NONE	0 Block of POTRERO AV	-122.4080	37.7692
140030006	WARRANTS	WARRANT ARRE...	Saturday	01/11/2014 08:00:00 AM +0000	08:30	MISSION	NONE	1000 Block of POTRERO AV	-122.4067	37.7568
140029275	WARRANTS	WARRANT ARRE...	Friday	01/10/2014 08:00:00 AM +0000	21:14	INGLESIDE	NONE	4100 Block of MISSION ST	-122.4300	37.7302
140029491	WARRANTS	WARRANT ARRE...	Friday	01/10/2014 08:00:00 AM +0000	23:24	INGLESIDE	NONE	MISSION ST / FRANCIS ST	-122.4334	37.7266
140011670	WARRANTS	WARRANT ARRE...	Saturday	01/04/2014 08:00:00 AM +0000	22:54	SOUTHERN	NONE	BRYANT ST / 8TH ST	-122.4070	37.7725
140011385	WARRANTS	WARRANT ARRE...	Saturday	01/04/2014 08:00:00 AM +0000	20:15	CENTRAL	NONE	700 Block of GEARY ST	-122.4153	37.7865
140000554	WARRANTS	WARRANT ARRE...	Wednesday	01/01/2014 08:00:00 AM +0000	04:13	MISSION	NONE	UTAH ST / 24TH ST	-122.4053	37.7531
140000344	WARRANTS	WARRANT ARRE...	Wednesday	01/01/2014 08:00:00 AM +0000	01:05	MISSION	NONE	18TH ST / GUERRERO ST	-122.4238	37.7616
131094451	WARRANTS	WARRANT ARRE...	Monday	12/30/2013 08:00:00 AM +0000	08:30	RICHMOND	NONE	ANZA ST / BLAKE ST	-122.4508	37.7806
131079641	WARRANTS	WARRANT ARRE...	Tuesday	12/24/2013 08:00:00 AM +0000	11:22	MISSION	NONE	1000 Block of POTRERO AV	-122.4067	37.7568
131079174	WARRANTS	WARRANT ARRE...	Tuesday	12/24/2013 08:00:00 AM +0000	08:03	MISSION	NONE	1000 Block of POTRERO AV	-122.4067	37.7568
131061262	WARRANTS	WARRANT ARRE...	Tuesday	12/17/2013 08:00:00 AM +0000	22:40	NORTHERN	NONE	500 Block of FRANKLIN ST	-122.4219	37.7795
131037489	WARRANTS	WARRANT ARRE...	Monday	12/09/2013 08:00:00 AM +0000	17:18	SOUTHERN	NONE	5TH ST / FOLSOM ST	-122.4034	37.7803
131033205	WARRANTS	WARRANT ARRE...	Sunday	12/08/2013 08:00:00 AM +0000	02:49	BAYVIEW	NONE	SHAFTER AV / JENNINGS ST	-122.3870	37.7289
131022985	WARRANTS	WARRANT ARRE...	Wednesday	12/04/2013 08:00:00 AM +0000	12:38	TENDERLOIN	NONE	EDDY ST / LEAVENWORTH ST	-122.4142	37.7837
131008151	WARRANTS	WARRANT ARRE...	Friday	11/29/2013 08:00:00 AM +0000	01:34	RICHMOND	NONE	2300 Block of GOLDEN GATE AV	-122.4488	37.7774
130996866	WARRANTS	WARRANT ARRE...	Sunday	11/24/2013 08:00:00 AM +0000	14:28	PARK	NONE	2100 Block of FULTON ST	-122.4519	37.7751
130974210	WARRANTS	WARRANT ARRE...	Sunday	11/17/2013 08:00:00 AM +0000	09:07	TENDERLOIN	NONE	0 Block of MASON ST	-122.4091	37.7837

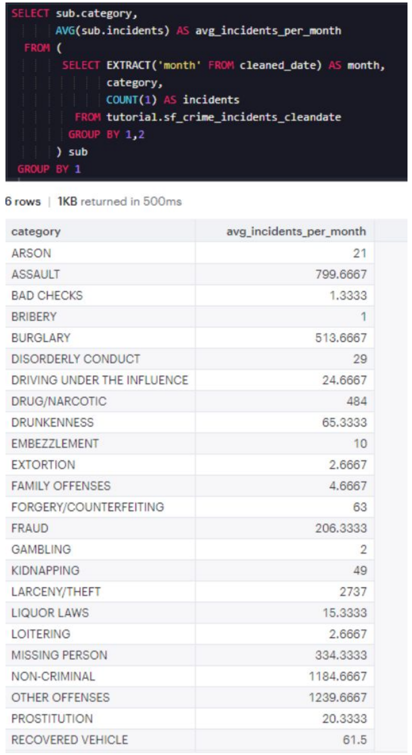
Practice Questions

Question-2: Write a query that displays the average monthly incidents for each category. Use tutorial.sf_crime_incidents_cleandate to solve this question.

Answer-2:

```
SELECT sub.category,
       AVG(sub.incidents) AS avg_incidents_per_month
  FROM (
        SELECT EXTRACT('month' FROM cleaned_date) AS month,
               category,
               COUNT(1) AS incidents
          FROM tutorial.sf_crime_incidents_cleandate
         GROUP BY 1,2
       ) sub
 GROUP BY 1
```

Practice Questions 2



```sql
SELECT sub.category,
       AVG(sub.incidents) AS avg_incidents_per_month
  FROM (
        SELECT EXTRACT('month' FROM cleaned_date) AS month,
               category,
               COUNT(1) AS incidents
          FROM tutorial.sf_crime_incidents_cleandate
         GROUP BY 1,2
       ) sub
 GROUP BY 1
```

6 rows | 1KB returned in 500ms

category	avg_incidents_per_month
ARSON	21
ASSAULT	799.6667
BAD CHECKS	1.3333
BRIBERY	1
BURGLARY	513.6667
DISORDERLY CONDUCT	29
DRIVING UNDER THE INFLUENCE	24.6667
DRUG/NARCOTIC	484
DRUNKENNESS	65.3333
EMBEZZLEMENT	10
EXTORTION	2.6667
FAMILY OFFENSES	4.6667
FORGERY/COUNTERFEITING	63
FRAUD	206.3333
GAMBLING	2
KIDNAPPING	49
LARCENY/THEFT	2737
LIQUOR LAWS	15.3333
LOITERING	2.6667
MISSING PERSON	334.3333
NON-CRIMINAL	1184.6667
OTHER OFFENSES	1239.6667
PROSTITUTION	20.3333
RECOVERED VEHICLE	61.5

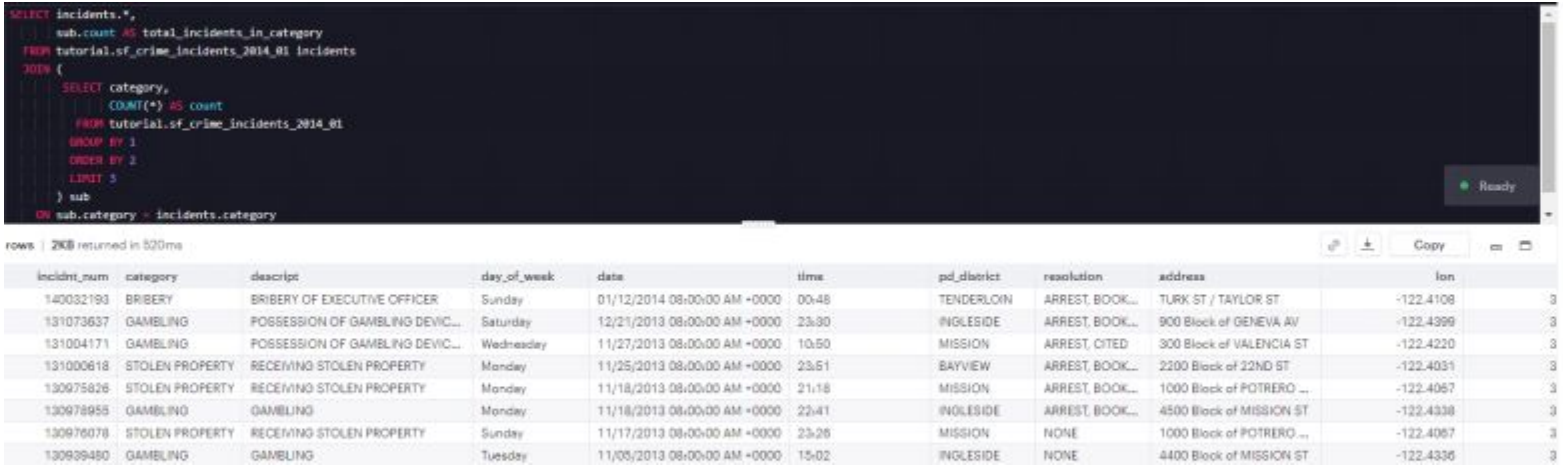
Practice Questions

Question-3: Create a query that returns all data from the three categories with the fewest reported events. Use sf_crime_incidents_2014_01 table.

Answer-3:

```
SELECT incidents.*,
       sub.count AS total_incidents_in_category
  FROM tutorial.sf_crime_incidents_2014_01 incidents
  JOIN (
        SELECT category,
               COUNT(*) AS count
         FROM tutorial.sf_crime_incidents_2014_01
        GROUP BY 1
        ORDER BY 2
        LIMIT 3
       ) sub
    ON sub.category = incidents.category
```

Practice Questions 3



```
SELECT incidents.*,
       sub.count AS total_incidents_in_category
  FROM tutorial.sf_crime_incidents_2014_01 incidents
  JOIN (
        SELECT category,
               COUNT(*) AS count
          FROM tutorial.sf_crime_incidents_2014_01
         GROUP BY 1
         ORDER BY 2
         LIMIT 5
       ) sub
    ON sub.category = incidents.category
```

rows | 2KB returned in 520ms

incidnt_num	category	descript	day_of_week	date	time	pd_district	resolution	address	lon	
140032193	BRIBERY	BRIBERY OF EXECUTIVE OFFICER	Sunday	01/12/2014 08:00:00 AM +0000	00:48	TENDERLOIN	ARREST, BOOK...	TURK ST / TAYLOR ST	-122.4108	3
131073637	GAMBLING	POSSESSION OF GAMBLING DEVIC...	Saturday	12/21/2013 08:00:00 AM +0000	23:30	INGLESIDE	ARREST, BOOK...	900 Block of GENEVA AV	-122.4399	3
131004171	GAMBLING	POSSESSION OF GAMBLING DEVIC...	Wednesday	11/27/2013 08:00:00 AM +0000	10:50	MISSION	ARREST, CITED	300 Block of VALENCIA ST	-122.4220	3
131000618	STOLEN PROPERTY	RECEIVING STOLEN PROPERTY	Monday	11/25/2013 08:00:00 AM +0000	23:51	BAYVIEW	ARREST, BOOK...	2200 Block of 22ND ST	-122.4031	3
130975826	STOLEN PROPERTY	RECEIVING STOLEN PROPERTY	Monday	11/18/2013 08:00:00 AM +0000	21:18	MISSION	ARREST, BOOK...	1000 Block of POTRERO ...	-122.4067	3
130978955	GAMBLING	GAMBLING	Monday	11/18/2013 08:00:00 AM +0000	22:41	INGLESIDE	ARREST, BOOK...	4500 Block of MISSION ST	-122.4338	3
130976078	STOLEN PROPERTY	RECEIVING STOLEN PROPERTY	Sunday	11/17/2013 08:00:00 AM +0000	23:26	MISSION	NONE	1000 Block of POTRERO ...	-122.4067	3
130939480	GAMBLING	GAMBLING	Tuesday	11/05/2013 08:00:00 AM +0000	15:02	INGLESIDE	NONE	4400 Block of MISSION ST	-122.4336	3

Practice Questions

Question-4: Write a query that counts the number of companies founded and acquired by quarter starting in Q1 2012. Create the aggregations in two separate queries, then join them. Use: tutorial.crunchbase_companies, tutorial.crunchbase_acquisitions tables.

Answer-4:

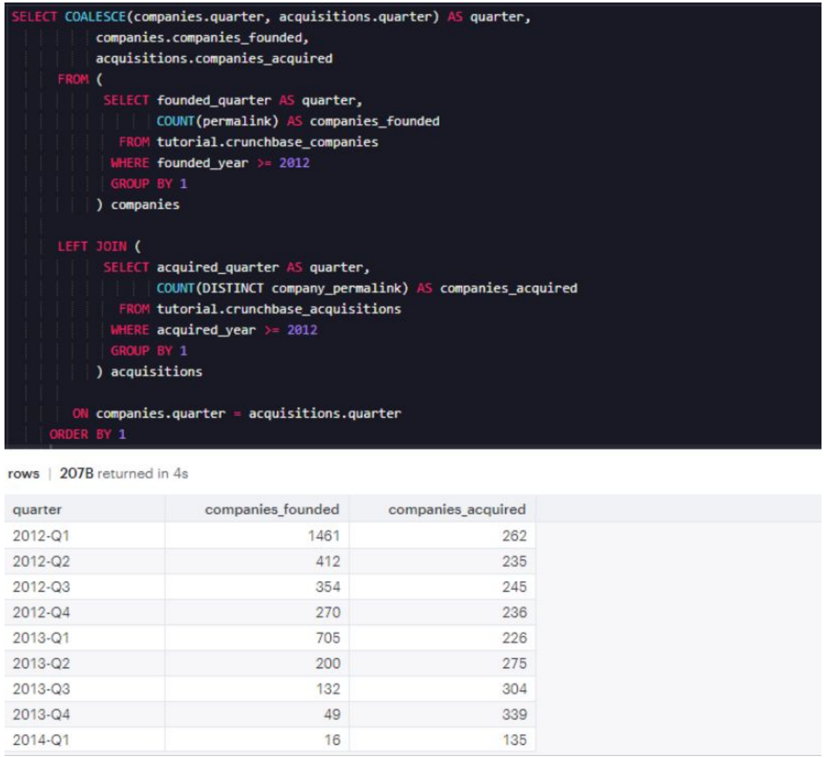
```
SELECT COALESCE(companies.quarter, acquisitions.quarter) AS quarter,
           companies.companies_founded,
           acquisitions.companies_acquired
      FROM (
            SELECT founded_quarter AS quarter,
                   COUNT(permalink) AS companies_founded
              FROM tutorial.crunchbase_companies
             WHERE founded_year >= 2012
             GROUP BY 1
           ) companies
```

Practice Questions

```
LEFT JOIN (
          SELECT acquired_quarter AS quarter,
                 COUNT(DISTINCT company_permalink) AS companies_acquired
            FROM tutorial.crunchbase_acquisitions
           WHERE acquired_year >= 2012
           GROUP BY 1
         ) acquisitions

      ON companies.quarter = acquisitions.quarter
   ORDER BY 1
```

Practice Questions 4



```
SELECT COALESCE(companies.quarter, acquisitions.quarter) AS quarter,
           companies.companies_founded,
           acquisitions.companies_acquired
      FROM (
            SELECT founded_quarter AS quarter,
                   COUNT(permalink) AS companies_founded
              FROM tutorial.crunchbase_companies
             WHERE founded_year >= 2012
             GROUP BY 1
           ) companies

      LEFT JOIN (
            SELECT acquired_quarter AS quarter,
                   COUNT(DISTINCT company_permalink) AS companies_acquired
              FROM tutorial.crunchbase_acquisitions
             WHERE acquired_year >= 2012
             GROUP BY 1
           ) acquisitions

        ON companies.quarter = acquisitions.quarter
     ORDER BY 1
```

rows | 207B returned in 4s

quarter	companies_founded	companies_acquired
2012-Q1	1461	262
2012-Q2	412	235
2012-Q3	354	245
2012-Q4	270	236
2013-Q1	705	226
2013-Q2	200	275
2013-Q3	132	304
2013-Q4	49	339
2014-Q1	16	135

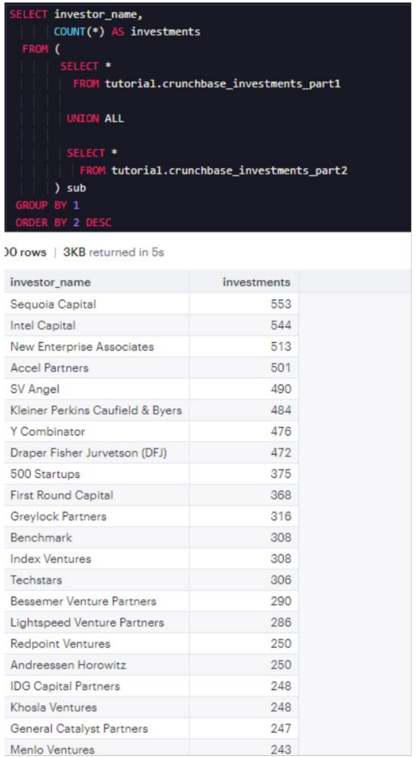
Practice Questions

Question-5: Write a query that ranks investors from the combined dataset above by the total number of investments they have made. Use: tutorial.crunchbase_investments_part1, tutorial.crunchbase_investments_part2 tables.

Answer-5:

```
SELECT investor_name,
       COUNT(*) AS investments
  FROM (
        SELECT *
          FROM tutorial.crunchbase_investments_part1
          UNION ALL
         SELECT *
           FROM tutorial.crunchbase_investments_part2
       ) sub
 GROUP BY 1
ORDER BY 2 DESC
```

Practice Questions 5



```
SELECT investor_name,
       COUNT(*) AS investments
  FROM (
        SELECT *
          FROM tutorial.crunchbase_investments_part1

         UNION ALL

         SELECT *
           FROM tutorial.crunchbase_investments_part2
       ) sub
 GROUP BY 1
 ORDER BY 2 DESC
```

0 rows | 3KB returned in 5s

investor_name	investments
Sequoia Capital	553
Intel Capital	544
New Enterprise Associates	513
Accel Partners	501
SV Angel	490
Kleiner Perkins Caufield & Byers	484
Y Combinator	476
Draper Fisher Jurvetson (DFJ)	472
500 Startups	375
First Round Capital	368
Greylock Partners	316
Benchmark	308
Index Ventures	308
Techstars	306
Bessemer Venture Partners	290
Lightspeed Venture Partners	286
Redpoint Ventures	250
Andreessen Horowitz	250
IDG Capital Partners	248
Khosla Ventures	248
General Catalyst Partners	247
Menlo Ventures	243

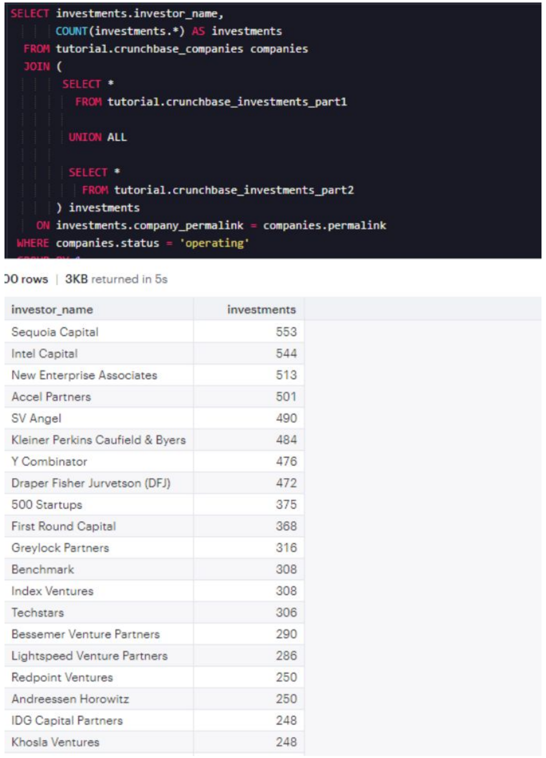
Practice Questions

Question-6: Write a query that ranks investors from the combined dataset above by the total number of investments they have made. Consider only the companies whose status is operating. Use: tutorial.crunchbase_investments_part1, tutorial.crunchbase_investments_part2 tables for investment. Use: tutorial.crunchbase_companies for status.

Answer-6:

```
SELECT investments.investor_name,
       COUNT(investments.*) AS investments
  FROM tutorial.crunchbase_companies companies
  JOIN (
        SELECT *
          FROM tutorial.crunchbase_investments_part1
         UNION ALL
         SELECT *
           FROM tutorial.crunchbase_investments_part2
       ) investments
    ON investments.company_permalink = companies.permalink
 WHERE companies.status = 'operating'
 GROUP BY 1
 ORDER BY 2 DESC
```

Practice Questions 6



```
SELECT investments.investor_name,
       COUNT(investments.*) AS investments
  FROM tutorial.crunchbase_companies companies
  JOIN (
        SELECT *
          FROM tutorial.crunchbase_investments_part1

         UNION ALL

        SELECT *
          FROM tutorial.crunchbase_investments_part2
       ) investments
    ON investments.company_permalink = companies.permalink
 WHERE companies.status = 'operating'
```

00 rows | 3KB returned in 5s

investor_name	investments
Sequoia Capital	553
Intel Capital	544
New Enterprise Associates	513
Accel Partners	501
SV Angel	490
Kleiner Perkins Caufield & Byers	484
Y Combinator	476
Draper Fisher Jurvetson (DFJ)	472
500 Startups	375
First Round Capital	368
Greylock Partners	316
Benchmark	308
Index Ventures	308
Techstars	306
Bessemer Venture Partners	290
Lightspeed Venture Partners	286
Redpoint Ventures	250
Andreessen Horowitz	250
IDG Capital Partners	248
Khosla Ventures	248

THANK YOU

In the next class we will study:

- Case statement and Common table Expression