Instacart Data analysis using AWS Athena, S3 and Glue

What is Athena?
Introduction:
Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run.

Athena is easy to use. Simply point to your data in Amazon S3, define the schema, and start querying using standard SQL. Most results are delivered within seconds. With Athena, there’s no need for complex ETL jobs to prepare your data for analysis. This makes it easy for anyone with SQL skills to quickly analyze large-scale datasets.

What is S3?
Introduction:
Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. This means customers of all sizes and industries can use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides easy-to-use management features so you can organize your data and configure finely-tuned access controls to meet your specific business, organizational, and compliance requirements. Amazon S3 is designed for 99.999999999% (11 9's) of durability, and stores data for millions of applications for companies all around the world.

What is IAM?
Introduction:
AWS Identity and Access Management (IAM) enables to manage access to AWS services and resources securely. Using IAM, one can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.

What is Glue crawler?
Introduction:
AWS Glue is a fully managed ETL (extract, transform, and load) service that makes it simple and cost-effective to categorize data, clean it, enrich it, and move it reliably between various data stores.
It consists of a central metadata repository known as the AWS Glue Data Catalog, an ETL engine that automatically generates Python or Scala code, and a flexible scheduler that handles dependency resolution, job monitoring, and retries. One can use AWS Glue to build a data warehouse to organize, cleanse, validate, and format data.

Data was loaded from local system storage to S3 bucket using the following steps:

1. Create bucket in S3 and rename it.
2. Upload or drag the files from local storage to newly created s3 bucket.

Steps to create table in Athena :

1. Click on create table in Athena
2. Create new database or choose from an existing one.
3. Provide the details such as Table name, location of input dataset, press next
4. Select the file extension, ie data format of file eg, csv,tsv etc
5. Add the names of columns of the selected file along with their datatype.
6. If partitions are needed select partition and then click create.
7. A new table will be created in Athena .

Alternatively , we can also write the sql query to create table, for example :

```
CREATE EXTERNAL TABLE IF NOT EXISTS insta.orders (
  `order_id` bigint,
  `user_id` bigint,
  `eval_set` string,
  `order_number` bigint,
  `order_dow` int,
  `order_hour_of_day` int,
  `days_since_prior_order` int
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe'
WITH SERDEPROPERTIES (
  'serialization.format' = ',',
  'field.delim' = ','
) LOCATION 's3://athenaswati12/orders/'
TBLPROPERTIES ('has_encrypted_data'='false');
```

Here table name is orders, taken from insta database

What is Instacart?

Introduction

Instacart is an online grocery delivery service company that works with local stores to deliver groceries to your door. Through it, customers can select groceries online through a web application and within a couple of hours(or a timeframe of one’s choice), the order gets delivered to the personal shopper.

Data Collection source:
Instacart Orders data has been taken from Kaggle. There were five datasets taken from the Kaggle website.

Data Dictionary :

1. Orders dataset:

#**order_id**: order identifier
#**user_id**: customer identifier
#**eval_set**: which evaluation set this order belongs in (see SET described below)
#**order_number**: the order sequence number for this user (1 = first, n = nth)
#**order_dow**: the day of the week the order was placed on
#**order_hour_of_day**: the hour of the day the order was placed on
#**days_since_prior**: days since the last order, capped at 30 (with NAs for order_number = 1)

This is my understanding of the dataset structure:

- users are identified by user_id in the orders csv file. Each row of the orders csv fil represents an order made by a user. Order are identified by order_id;
- Each order of a user is characterized by an order_number which specifies when it has been made with respect to the others of the same user;
- each order consists of a set of product each characterized by an add_to_cart_order feature representing the sequence in which they have been added to the cart in that order;

2. Products dataset :

#**product_id**: product identifier
#**product_name**: name of the product
#**aisle_id**: foreign key
#**department_id**: foreign key

3. Aisles dataset :
#**aisle_id**: aisle identifier
#**aisle**: the name of the aisle

4. Departments dataset :

#**department_id**: department identifier
#**department**: the name of the department

5. **order_products__SET** :

#**order_id**: foreign key
#**product_id**: foreign key
#**add_to_cart_order**: order in which each product was added to cart
#**reordered**: 1 if this product has been ordered by this user in the past, 0 otherwise
#where SET is one of the four following evaluation sets (eval_set in orders):

#"**prior**": orders prior to that users most recent order
#"***train***": training data supplied to participants
#"**test**": test data reserved for machine learning competitions

Data Interpretation

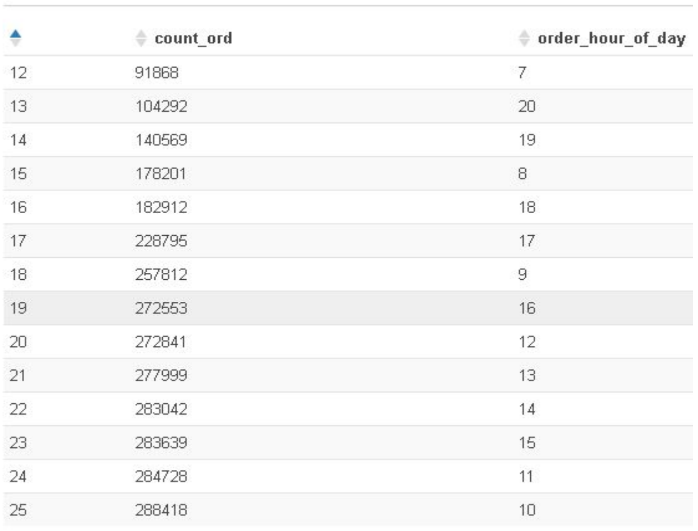
When do people order?

Interpretation:
People usually order during day time from 7 am to 4 pm mostly.

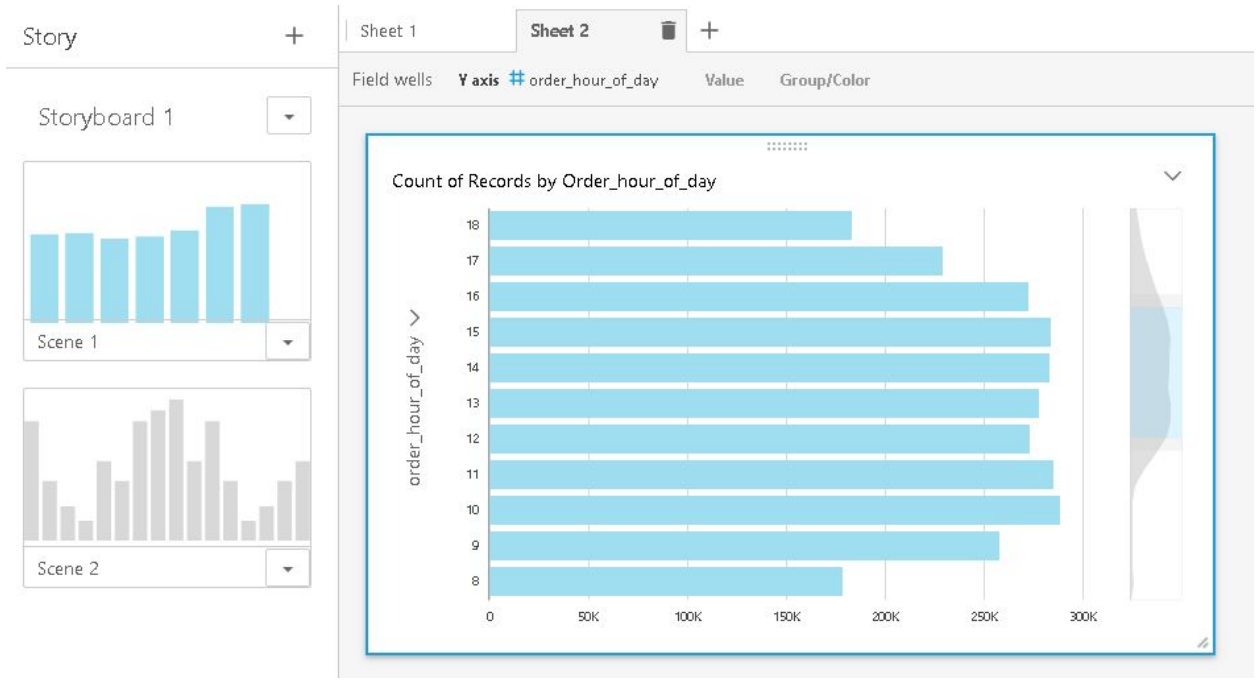
Query:

```
with temp as
(select count(user_id ) as count_ord,order_hour_of_day from orders group by
order_hour_of_day)
select  count_ord, order_hour_of_day from temp  order by count_ord
```

Result by Athena query:



	count_ord	order_hour_of_day
12	91868	7
13	104292	20
14	140569	19
15	178201	8
16	182912	18
17	228795	17
18	257812	9
19	272553	16
20	272841	12
21	277999	13
22	283042	14
23	283639	15
24	284728	11
25	288418	10

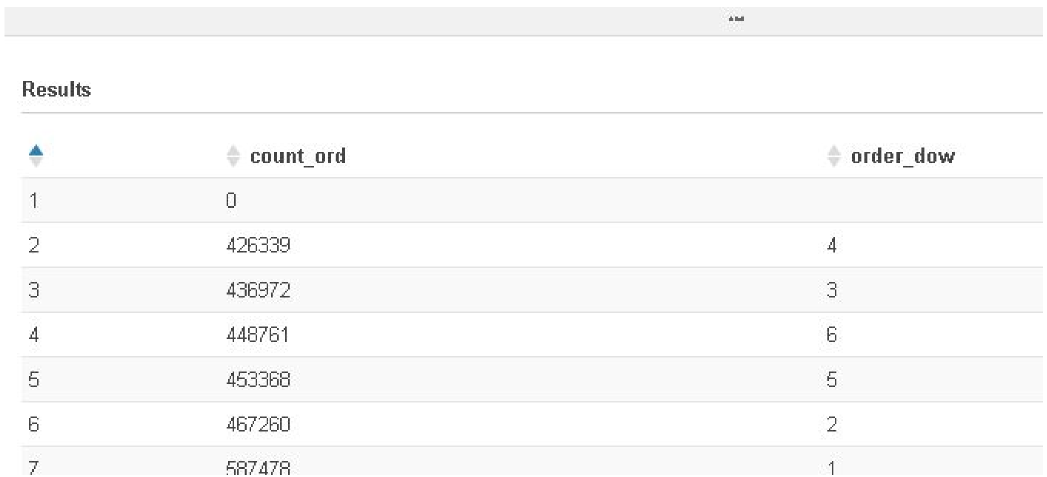


On which day of week , do people usually order?

Interpretation:
Most of the orders have been placed on days 0 and 1 which may probably be the weekends.

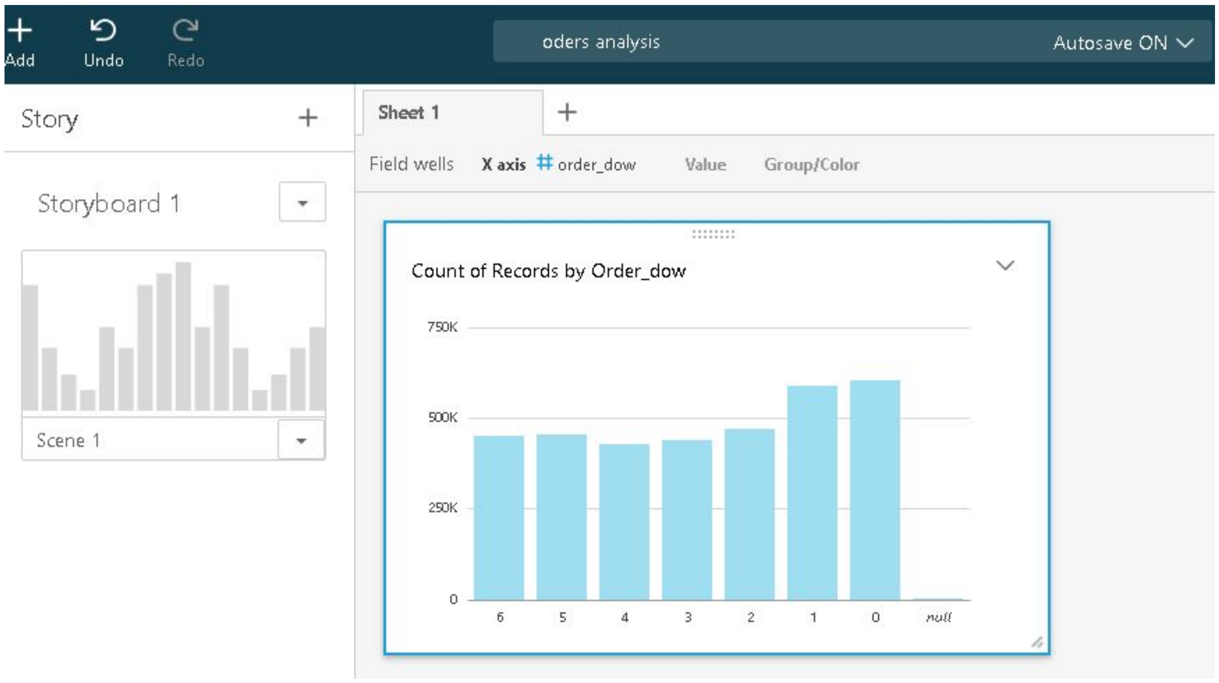
Query:
with temp as
(select count(user_id) as count_ord,order_dow from orders
group by order_dow) select count_ord, order_dow from temp order by count_ord

Result:



Results

	count_ord	order_dow
1	0	
2	426339	4
3	436972	3
4	448761	6
5	453368	5
6	467260	2
7	587478	1



Top Selling Product Departments

Interpretation:

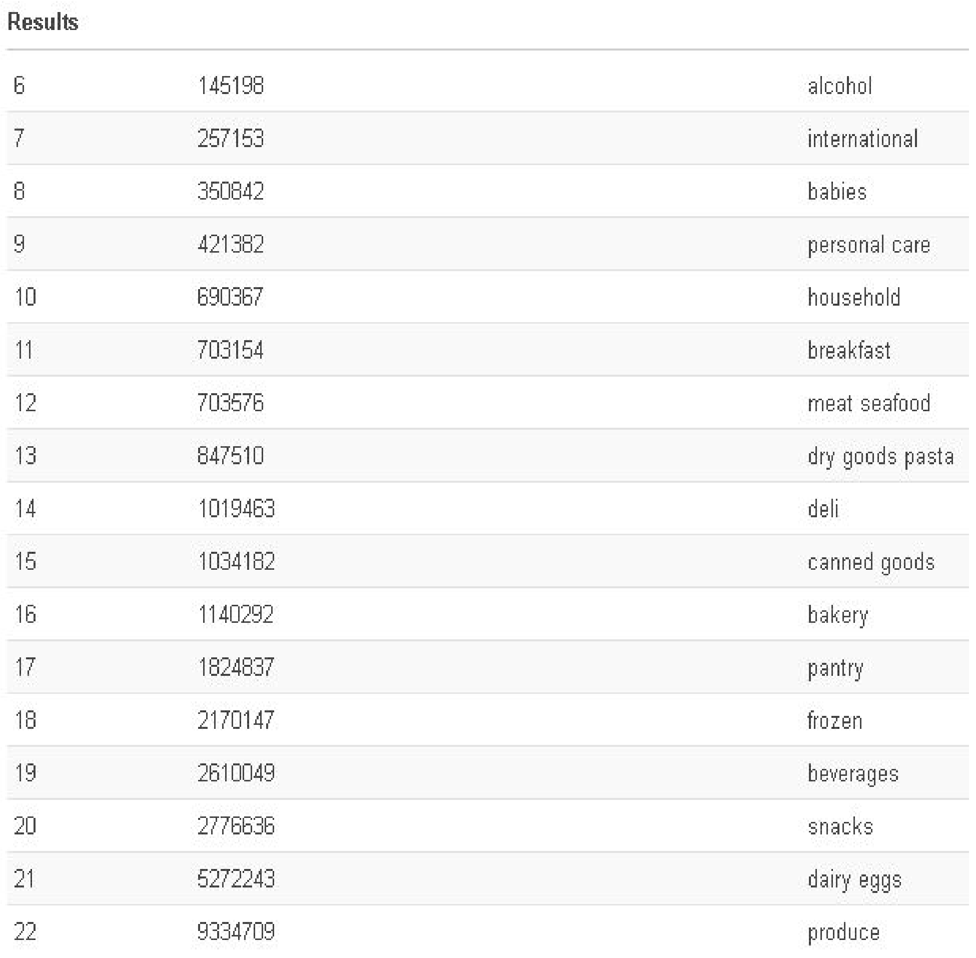
Most of the ordered products belong to produce department

Query:

```
with ord as (
select pp.product_id,pp.order_id,p.product_name,
p.aisle_id,p.department_id,p.department,pp.order_id, o.order_id,o.user_id,
pp.add_to_cart_order,pp.reordered from prod_dep  p full join prod_p pp
on p.product_id = pp.product_id
```

```
 full join orders  o on o.order_id = pp.order_id)
select count(product_name) as prod_count,department
from ord group by department   order by prod_count
```

Result:



Results

6	145198	alcohol
7	257153	international
8	350842	babies
9	421382	personal care
10	690367	household
11	703154	breakfast
12	703576	meat seafood
13	847510	dry goods pasta
14	1019463	deli
15	1034182	canned goods
16	1140292	bakery
17	1824837	pantry
18	2170147	frozen
19	2610049	beverages
20	2776636	snacks
21	5272243	dairy eggs
22	9334709	produce

Top Selling Products

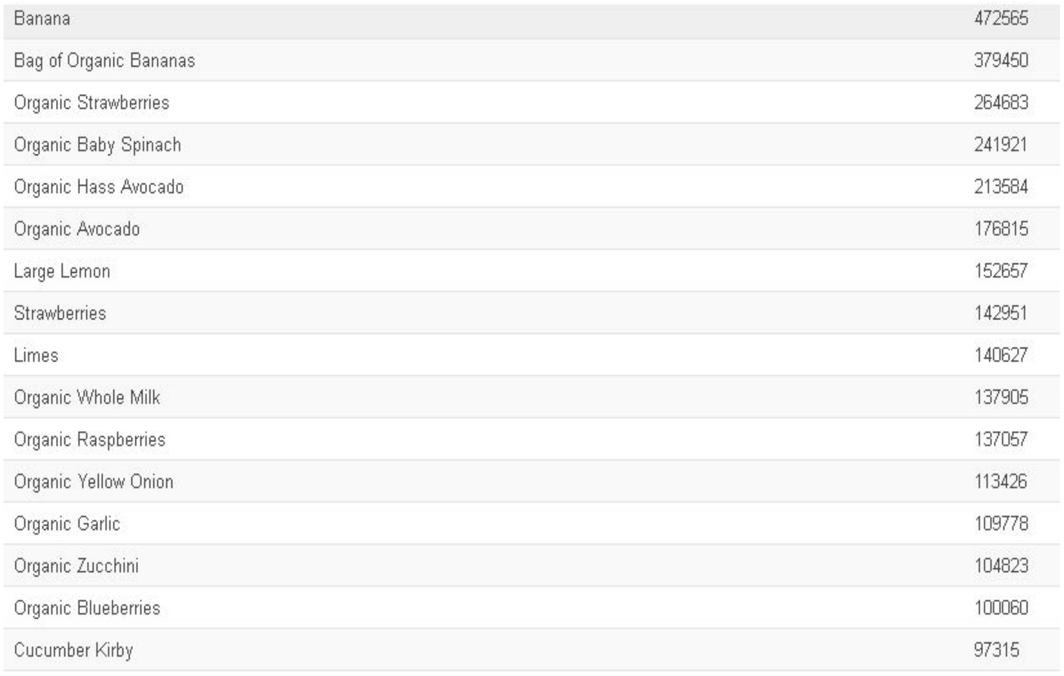
Interpretation:
Mostly ordered products are fruits like bananas , strawberries,etc.

Query:

```
with ord as (
select pp.product_id,pp.order_id,p.product_name, p.aisle_id, p.department_id,p.department,
pp.order_id,o.order_id,o.user_id,pp.add_to_cart_order, pp.reordered
from prod_dep  p full join prod_p pp
on p.product_id = pp.product_id
```

```
 full join orders  o on
  o.order_id = pp.order_id)
select product_name,count(*) as count
from ord
group by product_name
order by count desc
```

Result:



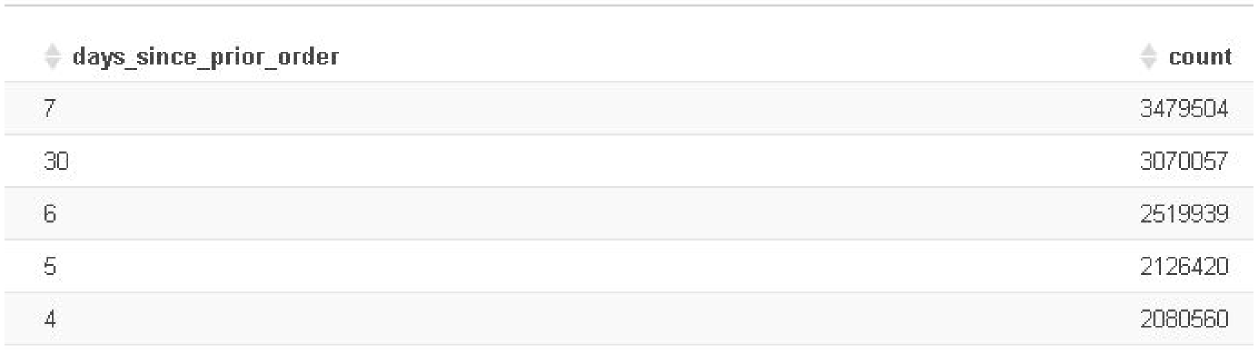
Banana	472565
Bag of Organic Bananas	379450
Organic Strawberries	264683
Organic Baby Spinach	241921
Organic Hass Avocado	213584
Organic Avocado	176815
Large Lemon	152657
Strawberries	142951
Limes	140627
Organic Whole Milk	137905
Organic Raspberries	137057
Organic Yellow Onion	113426
Organic Garlic	109778
Organic Zucchini	104823
Organic Blueberries	100060
Cucumber Kirby	97315

When do they order again?

Interpretation:
People usually order within 7 days or week after their previous order.

```
with ord as (
select pp.product_id,pp.order_id,p.product_name,
p.aisle_id,p.department_id,p.department,
pp.order_id,
O.order_id,o.days_since_prior_order,o.user_id,
pp.add_to_cart_order,pp.reordered
from prod_dep  p
full join prod_p pp
on p.product_id = pp.product_id
 full join orders  o on
  o.order_id = pp.order_id)
select days_since_prior_order,
count(product_id) as count
from ord
group by days_since_prior_order
order by days_since_prior_order
```

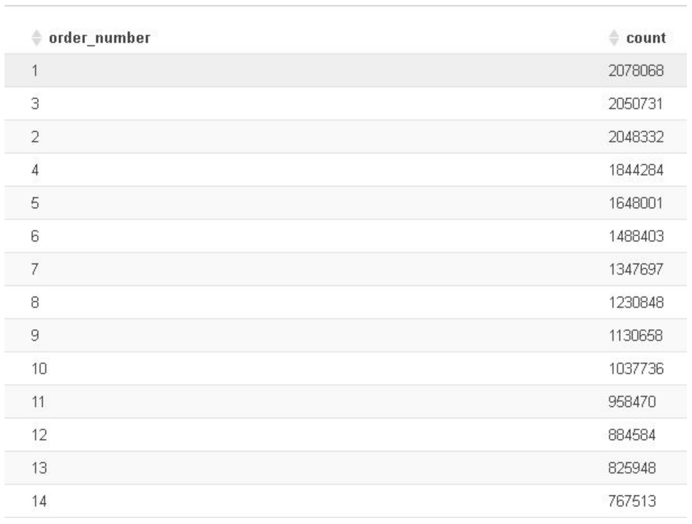
Result:



days_since_prior_order	count
7	3479504
30	3070057
6	2519939
5	2126420
4	2080560

Order Number

```
with ord as (
select pp.product_id,pp.order_id,p.product_name,
p.aisle_id,p.department_id,p.department,
pp.order_id,o.order_id,o.order_number,o.user_id,
pp.add_to_cart_order,pp.reordered
from prod_dep  p
full join prod_p pp
on p.product_id = pp.product_id
 full join orders  o on
  o.order_id = pp.order_id
  )
select order_number,count(order_number) as count
from ord
group by order_number
order by  count desc
```



order_number	count
1	2078068
3	2050731
2	2048332
4	1844284
5	1648001
6	1488403
7	1347697
8	1230848
9	1130658
10	1037736
11	958470
12	884584
13	825948
14	767513

Aisle Information

Interpretation:

Most of the ordered products belong to produce department and are from aisles fresh fruits and fresh vegetables.

Query:

```
with ord as (
select pp.product_id,pp.order_id, p.product_name,  p.aisle_id,a.aisle,p.department_id,
P.department,pp.order_id from prod_dep  p full join prod_p pp
on p.product_id = pp.product_id
 full join aisles  a on
  a.aisle_id = p.aisle_id
  )
select department,aisle,   count(product_name) as count
from ord
group by department,aisle   order by  count desc
```

Result:

department	aisle	count
produce	fresh fruits	3636924
produce	fresh vegetables	3362560
produce	packaged vegetables fruits	1702295
dairy eggs	yogurt	1396411
dairy eggs	packaged cheese	964567
dairy eggs	milk	834609
beverages	water seltzer sparkling water	830516
snacks	chips pretzels	682339
dairy eggs	soy lactosefree	630439
bakery	bread	563745
beverages	refrigerated	543171
frozen	frozen produce	517826
frozen	ice cream ice	482216