尚硅谷大数据技术之Hive SQL题库-中级

（作者：尚硅谷研究院）

版本：V1.0

# 第1章 环境准备

## 1.1 用户信息表

**1）表结构**

|  |  |  |
| --- | --- | --- |
| **user\_id(用户id)** | **gender(性别)** | **birthday(生日)** |
| **101** | 男 | 1990-01-01 |
| **102** | 女 | 1991-02-01 |
| **103** | 女 | 1992-03-01 |
| **104** | 男 | 1993-04-01 |

**2）建表语句**

hive>

DROP TABLE IF EXISTS user\_info;

create table user\_info(

`user\_id` string COMMENT '用户id',

`gender` string COMMENT '性别',

`birthday` string COMMENT '生日'

) COMMENT '用户信息表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table user\_info

values ('101', '男', '1990-01-01'),

('102', '女', '1991-02-01'),

('103', '女', '1992-03-01'),

('104', '男', '1993-04-01'),

('105', '女', '1994-05-01'),

('106', '男', '1995-06-01'),

('107', '女', '1996-07-01'),

('108', '男', '1997-08-01'),

('109', '女', '1998-09-01'),

('1010', '男', '1999-10-01');

## 1.2 商品信息表

**1）表结构**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **sku\_id**  **(商品id)** | **name**  **(商品名称)** | **category\_id**  **(分类id)** | **from\_date**  **(上架日期)** | **price**  **(商品价格)** |
| **1** | **xiaomi 10** | **1** | **2020-01-01** | **2000** |
| **6** | **洗碗机** | **2** | **2020-02-01** | **2000** |
| **9** | **自行车** | **3** | **2020-01-01** | **1000** |

**2）建表语句**

hive>

DROP TABLE IF EXISTS sku\_info;

CREATE TABLE sku\_info(

`sku\_id` string COMMENT '商品id',

`name` string COMMENT '商品名称',

`category\_id` string COMMENT '所属分类id',

`from\_date` string COMMENT '上架日期',

`price` double COMMENT '商品单价'

) COMMENT '商品属性表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table sku\_info

values ('1', 'xiaomi 10', '1', '2020-01-01', 2000),

('2', '手机壳', '1', '2020-02-01', 10),

('3', 'apple 12', '1', '2020-03-01', 5000),

('4', 'xiaomi 13', '1', '2020-04-01', 6000),

('5', '破壁机', '2', '2020-01-01', 500),

('6', '洗碗机', '2', '2020-02-01', 2000),

('7', '热水壶', '2', '2020-03-01', 100),

('8', '微波炉', '2', '2020-04-01', 600),

('9', '自行车', '3', '2020-01-01', 1000),

('10', '帐篷', '3', '2020-02-01', 100),

('11', '烧烤架', '3', '2020-02-01', 50),

('12', '遮阳伞', '3', '2020-03-01', 20);

## 1.3 商品分类信息表

**1）表结构**

|  |  |
| --- | --- |
| **category\_id(分类id)** | **category\_name(分类名称)** |
| **1** | 数码 |
| **2** | 厨卫 |
| **3** | 户外 |

**2）建表语句**

hive>

DROP TABLE IF EXISTS category\_info;

create table category\_info(

`category\_id` string,

`category\_name` string

) COMMENT '品类表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table category\_info

values ('1','数码'),

('2','厨卫'),

('3','户外');

## 1.4 订单信息表

**1）表结构**

|  |  |  |  |
| --- | --- | --- | --- |
| **order\_id**  **(订单id)** | **user\_id**  **(用户id)** | **create\_date**  **(下单日期)** | **total\_amount**  **(订单金额)** |
| **1** | 101 | 2021-09-30 | 29000.00 |
| **10** | 103 | 2020-10-02 | 28000.00 |

**2）建表语句**

hive>

DROP TABLE IF EXISTS order\_info;

create table order\_info(

`order\_id` string COMMENT '订单id',

`user\_id` string COMMENT '用户id',

`create\_date` string COMMENT '下单日期',

`total\_amount` decimal(16, 2) COMMENT '订单总金额'

) COMMENT '订单表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table order\_info

values ('1', '101', '2021-09-27', 29000.00),

('2', '101', '2021-09-28', 70500.00),

('3', '101', '2021-09-29', 43300.00),

('4', '101', '2021-09-30', 860.00),

('5', '102', '2021-10-01', 46180.00),

('6', '102', '2021-10-01', 50000.00),

('7', '102', '2021-10-01', 75500.00),

('8', '102', '2021-10-02', 6170.00),

('9', '103', '2021-10-02', 18580.00),

('10', '103', '2021-10-02', 28000.00),

('11', '103', '2021-10-02', 23400.00),

('12', '103', '2021-10-03', 5910.00),

('13', '104', '2021-10-03', 13000.00),

('14', '104', '2021-10-03', 69500.00),

('15', '104', '2021-10-03', 2000.00),

('16', '104', '2021-10-03', 5380.00),

('17', '105', '2021-10-04', 6210.00),

('18', '105', '2021-10-04', 68000.00),

('19', '105', '2021-10-04', 43100.00),

('20', '105', '2021-10-04', 2790.00),

('21', '106', '2021-10-04', 9390.00),

('22', '106', '2021-10-05', 58000.00),

('23', '106', '2021-10-05', 46600.00),

('24', '106', '2021-10-05', 5160.00),

('25', '107', '2021-10-05', 55350.00),

('26', '107', '2021-10-05', 14500.00),

('27', '107', '2021-10-06', 47400.00),

('28', '107', '2021-10-06', 6900.00),

('29', '108', '2021-10-06', 56570.00),

('30', '108', '2021-10-06', 44500.00),

('31', '108', '2021-10-07', 50800.00),

('32', '108', '2021-10-07', 3900.00),

('33', '109', '2021-10-07', 41480.00),

('34', '109', '2021-10-07', 88000.00),

('35', '109', '2020-10-08', 15000.00),

('36', '109', '2020-10-08', 9020.00),

('37', '1010', '2020-10-08', 9260.00),

('38', '1010', '2020-10-08', 12000.00),

('39', '1010', '2020-10-08', 23900.00),

('40', '1010', '2020-10-08', 6790.00);

## 1.5 订单明细表

**1）表结构**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **order\_detail\_id**  **(订单明细id)** | **order\_id**  **(订单id)** | **sku\_id**  **(商品id)** | **create\_date**  **(下单日期)** | **price**  **(商品单价)** | **sku\_num**  **(商品件数)** |
| **1** | **1** | **1** | **2021-09-30** | **2000.00** | **2** |
| **2** | **1** | **3** | **2021-09-30** | **5000.00** | **5** |
| **22** | **10** | **4** | **2020-10-02** | **6000.00** | **1** |
| **23** | **10** | **5** | **2020-10-02** | **500.00** | **24** |
| **24** | **10** | **6** | **2020-10-02** | **2000.00** | **5** |

**2）建表语句**

hive>

DROP TABLE IF EXISTS order\_detail;

CREATE TABLE order\_detail

(

`order\_detail\_id` string COMMENT '订单明细id',

`order\_id` string COMMENT '订单id',

`sku\_id` string COMMENT '商品id',

`create\_date` string COMMENT '下单日期',

`price` decimal(16, 2) COMMENT '下单时的商品单价',

`sku\_num` int COMMENT '下单商品件数'

) COMMENT '订单明细表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

INSERT overwrite table order\_detail

values ('1', '1', '1', '2021-09-27', 2000.00, 2),

('2', '1', '3', '2021-09-27', 5000.00, 5),

('3', '2', '4', '2021-09-28', 6000.00, 9),

('4', '2', '5', '2021-09-28', 500.00, 33),

('5', '3', '7', '2021-09-29', 100.00, 37),

('6', '3', '8', '2021-09-29', 600.00, 46),

('7', '3', '9', '2021-09-29', 1000.00, 12),

('8', '4', '12', '2021-09-30', 20.00, 43),

('9', '5', '1', '2021-10-01', 2000.00, 8),

('10', '5', '2', '2021-10-01', 10.00, 18),

('11', '5', '3', '2021-10-01', 5000.00, 6),

('12', '6', '4', '2021-10-01', 6000.00, 8),

('13', '6', '6', '2021-10-01', 2000.00, 1),

('14', '7', '7', '2021-10-01', 100.00, 17),

('15', '7', '8', '2021-10-01', 600.00, 48),

('16', '7', '9', '2021-10-01', 1000.00, 45),

('17', '8', '10', '2021-10-02', 100.00, 48),

('18', '8', '11', '2021-10-02', 50.00, 15),

('19', '8', '12', '2021-10-02', 20.00, 31),

('20', '9', '1', '2021-09-30', 2000.00, 9),

('21', '9', '2', '2021-10-02', 10.00, 5800),

('22', '10', '4', '2021-10-02', 6000.00, 1),

('23', '10', '5', '2021-10-02', 500.00, 24),

('24', '10', '6', '2021-10-02', 2000.00, 5),

('25', '11', '8', '2021-10-02', 600.00, 39),

('26', '12', '10', '2021-10-03', 100.00, 47),

('27', '12', '11', '2021-10-03', 50.00, 19),

('28', '12', '12', '2021-10-03', 20.00, 13000),

('29', '13', '1', '2021-10-03', 2000.00, 4),

('30', '13', '3', '2021-10-03', 5000.00, 1),

('31', '14', '4', '2021-10-03', 6000.00, 5),

('32', '14', '5', '2021-10-03', 500.00, 47),

('33', '14', '6', '2021-10-03', 2000.00, 8),

('34', '15', '7', '2021-10-03', 100.00, 20),

('35', '16', '10', '2021-10-03', 100.00, 22),

('36', '16', '11', '2021-10-03', 50.00, 42),

('37', '16', '12', '2021-10-03', 20.00, 7400),

('38', '17', '1', '2021-10-04', 2000.00, 3),

('39', '17', '2', '2021-10-04', 10.00, 21),

('40', '18', '4', '2021-10-04', 6000.00, 8),

('41', '18', '5', '2021-10-04', 500.00, 28),

('42', '18', '6', '2021-10-04', 2000.00, 3),

('43', '19', '7', '2021-10-04', 100.00, 55),

('44', '19', '8', '2021-10-04', 600.00, 11),

('45', '19', '9', '2021-10-04', 1000.00, 31),

('46', '20', '11', '2021-10-04', 50.00, 45),

('47', '20', '12', '2021-10-04', 20.00, 27),

('48', '21', '1', '2021-10-04', 2000.00, 2),

('49', '21', '2', '2021-10-04', 10.00, 39),

('50', '21', '3', '2021-10-04', 5000.00, 1),

('51', '22', '4', '2021-10-05', 6000.00, 8),

('52', '22', '5', '2021-10-05', 500.00, 20),

('53', '23', '7', '2021-10-05', 100.00, 58),

('54', '23', '8', '2021-10-05', 600.00, 18),

('55', '23', '9', '2021-10-05', 1000.00, 30),

('56', '24', '10', '2021-10-05', 100.00, 27),

('57', '24', '11', '2021-10-05', 50.00, 28),

('58', '24', '12', '2021-10-05', 20.00, 53),

('59', '25', '1', '2021-10-05', 2000.00, 5),

('60', '25', '2', '2021-10-05', 10.00, 35),

('61', '25', '3', '2021-10-05', 5000.00, 9),

('62', '26', '4', '2021-10-05', 6000.00, 1),

('63', '26', '5', '2021-10-05', 500.00, 13),

('64', '26', '6', '2021-10-05', 2000.00, 1),

('65', '27', '7', '2021-10-06', 100.00, 30),

('66', '27', '8', '2021-10-06', 600.00, 19),

('67', '27', '9', '2021-10-06', 1000.00, 33),

('68', '28', '10', '2021-10-06', 100.00, 37),

('69', '28', '11', '2021-10-06', 50.00, 46),

('70', '28', '12', '2021-10-06', 20.00, 45),

('71', '29', '1', '2021-10-06', 2000.00, 8),

('72', '29', '2', '2021-10-06', 10.00, 57),

('73', '29', '3', '2021-10-06', 5000.00, 8),

('74', '30', '4', '2021-10-06', 6000.00, 3),

('75', '30', '5', '2021-10-06', 500.00, 33),

('76', '30', '6', '2021-10-06', 2000.00, 5),

('77', '31', '8', '2021-10-07', 600.00, 13),

('78', '31', '9', '2021-10-07', 1000.00, 43),

('79', '32', '10', '2021-10-07', 100.00, 24),

('80', '32', '11', '2021-10-07', 50.00, 30),

('81', '33', '1', '2021-10-07', 2000.00, 8),

('82', '33', '2', '2021-10-07', 10.00, 48),

('83', '33', '3', '2021-10-07', 5000.00, 5),

('84', '34', '4', '2021-10-07', 6000.00, 10),

('85', '34', '5', '2021-10-07', 500.00, 44),

('86', '34', '6', '2021-10-07', 2000.00, 3),

('87', '35', '8', '2020-10-08', 600.00, 25),

('88', '36', '10', '2020-10-08', 100.00, 57),

('89', '36', '11', '2020-10-08', 50.00, 44),

('90', '36', '12', '2020-10-08', 20.00, 56),

('91', '37', '1', '2020-10-08', 2000.00, 2),

('92', '37', '2', '2020-10-08', 10.00, 26),

('93', '37', '3', '2020-10-08', 5000.00, 1),

('94', '38', '6', '2020-10-08', 2000.00, 6),

('95', '39', '7', '2020-10-08', 100.00, 35),

('96', '39', '8', '2020-10-08', 600.00, 34),

('97', '40', '10', '2020-10-08', 100.00, 37),

('98', '40', '11', '2020-10-08', 50.00, 51),

('99', '40', '12', '2020-10-08', 20.00, 27);

## 1.6 登录明细表

**1）表结构**

|  |  |  |  |
| --- | --- | --- | --- |
| user\_id(用户id) | ip\_address(ip地址) | login\_ts(登录时间) | logout\_ts(登出时间) |
| **101** | 180.149.130.161 | 2021-09-21 08:00:00 | 2021-09-27 08:30:00 |
| **102** | 120.245.11.2 | 2021-09-22 09:00:00 | 2021-09-27 09:30:00 |
| **103** | 27.184.97.3 | 2021-09-23 10:00:00 | 2021-09-27 10:30:00 |

**2）建表语句**

hive>

DROP TABLE IF EXISTS user\_login\_detail;

CREATE TABLE user\_login\_detail

(

`user\_id` string comment '用户id',

`ip\_address` string comment 'ip地址',

`login\_ts` string comment '登录时间',

`logout\_ts` string comment '登出时间'

) COMMENT '用户登录明细表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

INSERT overwrite table user\_login\_detail

VALUES ('101', '180.149.130.161', '2021-09-21 08:00:00', '2021-09-27 08:30:00'),

('101', '180.149.130.161', '2021-09-27 08:00:00', '2021-09-27 08:30:00'),

('101', '180.149.130.161', '2021-09-28 09:00:00', '2021-09-28 09:10:00'),

('101', '180.149.130.161', '2021-09-29 13:30:00', '2021-09-29 13:50:00'),

('101', '180.149.130.161', '2021-09-30 20:00:00', '2021-09-30 20:10:00'),

('102', '120.245.11.2', '2021-09-22 09:00:00', '2021-09-27 09:30:00'),

('102', '120.245.11.2', '2021-10-01 08:00:00', '2021-10-01 08:30:00'),

('102', '180.149.130.174', '2021-10-01 07:50:00', '2021-10-01 08:20:00'),

('102', '120.245.11.2', '2021-10-02 08:00:00', '2021-10-02 08:30:00'),

('103', '27.184.97.3', '2021-09-23 10:00:00', '2021-09-27 10:30:00'),

('103', '27.184.97.3', '2021-10-03 07:50:00', '2021-10-03 09:20:00'),

('104', '27.184.97.34', '2021-09-24 11:00:00', '2021-09-27 11:30:00'),

('104', '27.184.97.34', '2021-10-03 07:50:00', '2021-10-03 08:20:00'),

('104', '27.184.97.34', '2021-10-03 08:50:00', '2021-10-03 10:20:00'),

('104', '120.245.11.89', '2021-10-03 08:40:00', '2021-10-03 10:30:00'),

('105', '119.180.192.212', '2021-10-04 09:10:00', '2021-10-04 09:30:00'),

('106', '119.180.192.66', '2021-10-04 08:40:00', '2021-10-04 10:30:00'),

('106', '119.180.192.66', '2021-10-05 21:50:00', '2021-10-05 22:40:00'),

('107', '219.134.104.7', '2021-09-25 12:00:00', '2021-09-27 12:30:00'),

('107', '219.134.104.7', '2021-10-05 22:00:00', '2021-10-05 23:00:00'),

('107', '219.134.104.7', '2021-10-06 09:10:00', '2021-10-06 10:20:00'),

('107', '27.184.97.46', '2021-10-06 09:00:00', '2021-10-06 10:00:00'),

('108', '101.227.131.22', '2021-10-06 09:00:00', '2021-10-06 10:00:00'),

('108', '101.227.131.22', '2021-10-06 22:00:00', '2021-10-06 23:00:00'),

('109', '101.227.131.29', '2021-09-26 13:00:00', '2021-09-27 13:30:00'),

('109', '101.227.131.29', '2021-10-06 08:50:00', '2021-10-06 10:20:00'),

('109', '101.227.131.29', '2021-10-08 09:00:00', '2021-10-08 09:10:00'),

('1010', '119.180.192.10', '2021-09-27 14:00:00', '2021-09-27 14:30:00'),

('1010', '119.180.192.10', '2021-10-09 08:50:00', '2021-10-09 10:20:00');

## 1.7 商品价格变更明细表

**1）表结构**

|  |  |  |
| --- | --- | --- |
| **sku\_id(商品id)** | **new\_price(本次变更之后的价格)** | **change\_date(变更日期)** |
| **1** | 1900.00 | 2021-09-25 |
| **1** | 2000.00 | 2021-09-26 |
| **2** | 80.00 | 2021-09-29 |
| **2** | 10.00 | 2021-09-30 |

**2）建表语句**

hive>

DROP TABLE IF EXISTS sku\_price\_modify\_detail;

CREATE TABLE sku\_price\_modify\_detail

(

`sku\_id` string comment '商品id',

`new\_price` decimal(16, 2) comment '更改后的价格',

`change\_date` string comment '变动日期'

) COMMENT '商品价格变更明细表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table sku\_price\_modify\_detail

values ('1', 1900, '2021-09-25'),

('1', 2000, '2021-09-26'),

('2', 80, '2021-09-29'),

('2', 10, '2021-09-30'),

('3', 4999, '2021-09-25'),

('3', 5000, '2021-09-26'),

('4', 5600, '2021-09-26'),

('4', 6000, '2021-09-27'),

('5', 490, '2021-09-27'),

('5', 500, '2021-09-28'),

('6', 1988, '2021-09-30'),

('6', 2000, '2021-10-01'),

('7', 88, '2021-09-28'),

('7', 100, '2021-09-29'),

('8', 800, '2021-09-28'),

('8', 600, '2021-09-29'),

('9', 1100, '2021-09-27'),

('9', 1000, '2021-09-28'),

('10', 90, '2021-10-01'),

('10', 100, '2021-10-02'),

('11', 66, '2021-10-01'),

('11', 50, '2021-10-02'),

('12', 35, '2021-09-28'),

('12', 20, '2021-09-29');

## 1.8 配送信息表

**1）表结构**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **delivery\_id**  **（运单id）** | **order\_id**  **（订单id）** | **user\_id**  **（用户id）** | **order\_date**  **（下单日期）** | **custom\_date**  **（期望配送日期）** |
| **1** | 1 | 101 | 2021-09-27 | 2021-09-29 |
| **2** | 2 | 101 | 2021-09-28 | 2021-09-28 |
| **3** | 3 | 101 | 2021-09-29 | 2021-09-30 |

**2）建表语句**

hive>

DROP TABLE IF EXISTS delivery\_info;

CREATE TABLE delivery\_info

(

`delivery\_id` string comment '配送单id',

`order\_id` string comment '订单id',

`user\_id` string comment '用户id',

`order\_date` string comment '下单日期',

`custom\_date` string comment '期望配送日期'

) COMMENT '邮寄信息表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table delivery\_info

values ('1', '1', '101', '2021-09-27', '2021-09-29'),

('2', '2', '101', '2021-09-28', '2021-09-28'),

('3', '3', '101', '2021-09-29', '2021-09-30'),

('4', '4', '101', '2021-09-30', '2021-10-01'),

('5', '5', '102', '2021-10-01', '2021-10-01'),

('6', '6', '102', '2021-10-01', '2021-10-01'),

('7', '7', '102', '2021-10-01', '2021-10-03'),

('8', '8', '102', '2021-10-02', '2021-10-02'),

('9', '9', '103', '2021-10-02', '2021-10-03'),

('10', '10', '103', '2021-10-02', '2021-10-04'),

('11', '11', '103', '2021-10-02', '2021-10-02'),

('12', '12', '103', '2021-10-03', '2021-10-03'),

('13', '13', '104', '2021-10-03', '2021-10-04'),

('14', '14', '104', '2021-10-03', '2021-10-04'),

('15', '15', '104', '2021-10-03', '2021-10-03'),

('16', '16', '104', '2021-10-03', '2021-10-03'),

('17', '17', '105', '2021-10-04', '2021-10-04'),

('18', '18', '105', '2021-10-04', '2021-10-06'),

('19', '19', '105', '2021-10-04', '2021-10-06'),

('20', '20', '105', '2021-10-04', '2021-10-04'),

('21', '21', '106', '2021-10-04', '2021-10-04'),

('22', '22', '106', '2021-10-05', '2021-10-05'),

('23', '23', '106', '2021-10-05', '2021-10-05'),

('24', '24', '106', '2021-10-05', '2021-10-07'),

('25', '25', '107', '2021-10-05', '2021-10-05'),

('26', '26', '107', '2021-10-05', '2021-10-06'),

('27', '27', '107', '2021-10-06', '2021-10-06'),

('28', '28', '107', '2021-10-06', '2021-10-07'),

('29', '29', '108', '2021-10-06', '2021-10-06'),

('30', '30', '108', '2021-10-06', '2021-10-06'),

('31', '31', '108', '2021-10-07', '2021-10-09'),

('32', '32', '108', '2021-10-07', '2021-10-09'),

('33', '33', '109', '2021-10-07', '2021-10-08'),

('34', '34', '109', '2021-10-07', '2021-10-08'),

('35', '35', '109', '2021-10-08', '2021-10-10'),

('36', '36', '109', '2021-10-08', '2021-10-09'),

('37', '37', '1010', '2021-10-08', '2021-10-10'),

('38', '38', '1010', '2021-10-08', '2021-10-10'),

('39', '39', '1010', '2021-10-08', '2021-10-09'),

('40', '40', '1010', '2021-10-08', '2021-10-09');

## 1.9 好友关系表

**1）表结构**

|  |  |
| --- | --- |
| **user1\_id（用户1 id）** | **user2\_id（用户2 id）** |
| **101** | 1010 |
| **101** | 108 |
| **101** | 106 |

注：表中一行数据中的两个user\_id，表示两个用户互为好友。

**2）建表语句**

hive>

DROP TABLE IF EXISTS friendship\_info;

CREATE TABLE friendship\_info(

`user1\_id` string comment '用户1id',

`user2\_id` string comment '用户2id'

) COMMENT '用户关系表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table friendship\_info

values ('101', '1010'),

('101', '108'),

('101', '106'),

('101', '104'),

('101', '102'),

('102', '1010'),

('102', '108'),

('102', '106'),

('102', '104'),

('102', '102'),

('103', '1010'),

('103', '108'),

('103', '106'),

('103', '104'),

('103', '102'),

('104', '1010'),

('104', '108'),

('104', '106'),

('104', '104'),

('104', '102'),

('105', '1010'),

('105', '108'),

('105', '106'),

('105', '104'),

('105', '102'),

('106', '1010'),

('106', '108'),

('106', '106'),

('106', '104'),

('106', '102'),

('107', '1010'),

('107', '108'),

('107', '106'),

('107', '104'),

('107', '102'),

('108', '1010'),

('108', '108'),

('108', '106'),

('108', '104'),

('108', '102'),

('109', '1010'),

('109', '108'),

('109', '106'),

('109', '104'),

('109', '102'),

('1010', '1010'),

('1010', '108'),

('1010', '106'),

('1010', '104'),

('1010', '102');

## 1.10 收藏信息表

**1）表结构**

|  |  |  |
| --- | --- | --- |
| **user\_id(用户id)** | **sku\_id(商品id)** | **create\_date(收藏日期)** |
| **101** | 3 | 2021-09-23 |
| **101** | 12 | 2021-09-23 |
| **101** | 6 | 2021-09-25 |

**2）建表语句**

hive>

DROP TABLE IF EXISTS favor\_info;

CREATE TABLE favor\_info

(

`user\_id` string comment '用户id',

`sku\_id` string comment '商品id',

`create\_date` string comment '收藏日期'

) COMMENT '用户收藏表'

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

**3）数据装载**

hive>

insert overwrite table favor\_info

values ('101', '3', '2021-09-23'),

('101', '12', '2021-09-23'),

('101', '6', '2021-09-25'),

('101', '10', '2021-09-21'),

('101', '5', '2021-09-25'),

('102', '1', '2021-09-24'),

('102', '2', '2021-09-24'),

('102', '8', '2021-09-23'),

('102', '12', '2021-09-22'),

('102', '11', '2021-09-23'),

('102', '9', '2021-09-25'),

('102', '4', '2021-09-25'),

('102', '6', '2021-09-23'),

('102', '7', '2021-09-26'),

('103', '8', '2021-09-24'),

('103', '5', '2021-09-25'),

('103', '6', '2021-09-26'),

('103', '12', '2021-09-27'),

('103', '7', '2021-09-25'),

('103', '10', '2021-09-25'),

('103', '4', '2021-09-24'),

('103', '11', '2021-09-25'),

('103', '3', '2021-09-27'),

('104', '9', '2021-09-28'),

('104', '7', '2021-09-28'),

('104', '8', '2021-09-25'),

('104', '3', '2021-09-28'),

('104', '11', '2021-09-25'),

('104', '6', '2021-09-25'),

('104', '12', '2021-09-28'),

('105', '8', '2021-10-08'),

('105', '9', '2021-10-07'),

('105', '7', '2021-10-07'),

('105', '11', '2021-10-06'),

('105', '5', '2021-10-07'),

('105', '4', '2021-10-05'),

('105', '10', '2021-10-07'),

('106', '12', '2021-10-08'),

('106', '1', '2021-10-08'),

('106', '4', '2021-10-04'),

('106', '5', '2021-10-08'),

('106', '2', '2021-10-04'),

('106', '6', '2021-10-04'),

('106', '7', '2021-10-08'),

('107', '5', '2021-09-29'),

('107', '3', '2021-09-28'),

('107', '10', '2021-09-27'),

('108', '9', '2021-10-08'),

('108', '3', '2021-10-10'),

('108', '8', '2021-10-10'),

('108', '10', '2021-10-07'),

('108', '11', '2021-10-07'),

('109', '2', '2021-09-27'),

('109', '4', '2021-09-29'),

('109', '5', '2021-09-29'),

('109', '9', '2021-09-30'),

('109', '8', '2021-09-26'),

('1010', '2', '2021-09-29'),

('1010', '9', '2021-09-29'),

('1010', '1', '2021-10-01');

# 第2章 练习题

## 2.1 [课堂讲解]查询累积销量排名第二的商品

### 2.1.1 题目需求

查询订单明细表（order\_detail）中销量（下单件数）排名第二的商品id，如果不存在返回null，如果存在多个排名第二的商品则需要全部返回。期望结果如下：

|  |
| --- |
| **sku\_id** |
| **11** |

### 2.1.2 代码实现

hive>

select sku\_id

from (

select sku\_id

from (

select sku\_id,

order\_num,

dense\_rank() over (order by order\_num desc) rk

from (

select sku\_id,

sum(sku\_num) order\_num

from order\_detail

group by sku\_id

) t1

) t2

where rk = 2

) t3

right join --为保证，没有第二名的情况下，返回null

(

select 1

) t4

on 1 = 1;

## 2.2 [课堂讲解]查询至少连续三天下单的用户

### 2.2.1 题目需求

查询订单信息表(order\_info)中最少连续3天下单的用户id，期望结果如下：

|  |
| --- |
| **user\_id** |
| **101** |

### 2.2.2 代码实现

hive>

select distinct user\_id

from (

select user\_id

from (

select user\_id

, create\_date

, date\_sub(create\_date, row\_number() over (partition by user\_id order by create\_date)) flag

from (

select user\_id

, create\_date

from order\_info

group by user\_id, create\_date

) t1 -- 同一天可能多个用户下单，进行去重

) t2 -- 判断一串日期是否连续：若连续，用这个日期减去它的排名，会得到一个相同的结果

group by user\_id, flag

having count(flag) >= 3 -- 连续下单大于等于三天

) t3;

## 2.3 [课堂讲解]查询各品类销售商品的种类数及销量最高的商品

### 2.3.1 题目需求

从订单明细表(order\_detail)统计各品类销售出的商品种类数及累积销量最好的商品，期望结果如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **category\_id**  **(分类id)** | **category\_name**  **(分类名称)** | **sku\_id**  **(销量最好的商品id)** | **name**  **(商品名称)** | **order\_num**  **（销量最好的商品销量）** | **order\_cnt**  **(商品种类数量)** |
| **1** | 数码 | 2 | 手机壳 | 302 | 4 |
| **2** | 厨卫 | 8 | 微波炉 | 253 | 4 |
| **3** | 户外 | 12 | 遮阳伞 | 349 | 4 |

### 2.3.2 代码实现

hive>

select category\_id,

category\_name,

sku\_id,

name,

order\_num,

sku\_cnt

from (

select od.sku\_id,

sku.name,

sku.category\_id,

cate.category\_name,

order\_num,

rank() over (partition by sku.category\_id order by order\_num desc) rk,

count(distinct od.sku\_id) over (partition by sku.category\_id) sku\_cnt

from (

select sku\_id,

sum(sku\_num) order\_num

from order\_detail

group by sku\_id

) od

left join

sku\_info sku

on od.sku\_id = sku.sku\_id

left join

category\_info cate

on sku.category\_id = cate.category\_id

) t1

where rk = 1;

## 2.4 [课堂讲解]查询用户的累计消费金额及VIP等级

### 2.4.1 题目需求

从订单信息表(order\_info)中统计每个用户截止其每个下单日期的累积消费金额，以及每个用户在其每个下单日期的VIP等级。

用户vip等级根据累积消费金额计算，计算规则如下：

设累积消费总额为X，

若0=<X<10000,则vip等级为普通会员

若10000<=X<30000,则vip等级为青铜会员

若30000<=X<50000,则vip等级为白银会员

若50000<=X<80000,则vip为黄金会员

若80000<=X<100000,则vip等级为白金会员

若X>=100000,则vip等级为钻石会员

期望结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **user\_id**  **(用户id)** | **create\_date**  **(下单日期)** | **sum\_so\_far**  **(截至每个下单日期的累计下单金额)** | **vip\_level**  **(每个下单日期的VIP等级)** |
| **101** | 2021-09-27 | 29000.00 | 青铜会员 |
| **101** | 2021-09-28 | 99500.00 | 白金会员 |
| **101** | 2021-09-29 | 142800.00 | 钻石会员 |
| **101** | 2021-09-30 | 143660.00 | 钻石会员 |
| **102** | 2021-10-01 | 171680.00 | 钻石会员 |
| **102** | 2021-10-02 | 177850.00 | 钻石会员 |
| **103** | 2021-10-02 | 69980.00 | 黄金会员 |
| **103** | 2021-10-03 | 75890.00 | 黄金会员 |
| **104** | 2021-10-03 | 89880.00 | 白金会员 |
| **105** | 2021-10-04 | 120100.00 | 钻石会员 |
| **106** | 2021-10-04 | 9390.00 | 普通会员 |
| **106** | 2021-10-05 | 119150.00 | 钻石会员 |
| **107** | 2021-10-05 | 69850.00 | 黄金会员 |
| **107** | 2021-10-06 | 124150.00 | 钻石会员 |
| **108** | 2021-10-06 | 101070.00 | 钻石会员 |
| **108** | 2021-10-07 | 155770.00 | 钻石会员 |
| **109** | 2021-10-07 | 129480.00 | 钻石会员 |
| **109** | 2021-10-08 | 153500.00 | 钻石会员 |
| **1010** | 2021-10-08 | 51950.00 | 黄金会员 |

### 2.4.2 代码实现

hive>

select user\_id,

create\_date,

sum\_so\_far,

case

when sum\_so\_far >= 100000 then '钻石会员'

when sum\_so\_far >= 80000 then '白金会员'

when sum\_so\_far >= 50000 then '黄金会员'

when sum\_so\_far >= 30000 then '白银会员'

when sum\_so\_far >= 10000 then '青铜会员'

when sum\_so\_far >= 0 then '普通会员'

end vip\_level

from (

select user\_id,

create\_date,

sum(total\_amount\_per\_day) over (partition by user\_id order by create\_date) sum\_so\_far

from (

select user\_id,

create\_date,

sum(total\_amount) total\_amount\_per\_day

from order\_info

group by user\_id, create\_date

) t1

) t2;

## 2.5 [课堂讲解]查询首次下单后第二天连续下单的用户比率

### 2.5.1 题目需求

从订单信息表(order\_info)中查询首次下单后第二天仍然下单的用户占所有下单用户的比例，结果保留一位小数，使用百分数显示，期望结果如下：

|  |
| --- |
| **percentage** |
| **70.0%** |

### 2.5.2 代码实现

hive>

select concat(round(sum(if(datediff(buy\_date\_second, buy\_date\_first) = 1, 1, 0)) / count(\*) \* 100, 1), '%') percentage

from (

select user\_id,

min(create\_date) buy\_date\_first,

max(create\_date) buy\_date\_second

from (

select user\_id,

create\_date,

rank() over (partition by user\_id order by create\_date) rk

from (

select user\_id,

create\_date

from order\_info

group by user\_id, create\_date

) t1

) t2

where rk <= 2

group by user\_id

) t3;

## 2.6 每个商品销售首年的年份、销售数量和销售金额

### 2.6.1 题目需求

从订单明细表(order\_detail)统计每个商品销售首年的年份，销售数量和销售总额。

期望结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **sku\_id**  **（商品id）** | **year**  **（销售首年年份）** | **order\_num**  **（首年销量）** | **order\_amount**  **（首年销售金额）** |
| **1** | **2021** | **51** | **102000.00** |
| **2** | **2021** | **302** | **3020.00** |
| **3** | **2021** | **36** | **180000.00** |
| **4** | **2021** | **53** | **318000.00** |
| **5** | **2021** | **242** | **121000.00** |
| **6** | **2021** | **32** | **64000.00** |
| **7** | **2021** | **252** | **25200.00** |
| **8** | **2021** | **253** | **151800.00** |
| **9** | **2021** | **194** | **194000.00** |
| **10** | **2021** | **299** | **29900.00** |
| **11** | **2021** | **320** | **16000.00** |
| **12** | **2021** | **349** | **6980.00** |

### 2.6.2 代码实现

hive>

select sku\_id,

year(create\_date),

sum(sku\_num),

sum(price\*sku\_num)

from (

select order\_id,

sku\_id,

price,

sku\_num,

create\_date,

rank() over (partition by sku\_id order by year(create\_date)) rk

from order\_detail

) t1

where rk = 1

group by sku\_id,year(create\_date);

## 2.7 筛选去年总销量小于100的商品

### 2.7.1 题目需求

从订单明细表(order\_detail)中筛选出去年总销量小于100的商品及其销量，假设今天的日期是2022-01-10，不考虑上架时间小于一个月的商品，期望结果如下：

|  |  |  |
| --- | --- | --- |
| **sku\_id**  **（商品id）** | **name**  **（商品名称）** | **order\_num**  **（销量）** |
| **1** | xiaomi 10 | 51 |
| **3** | apple 12 | 36 |
| **4** | xiaomi 13 | 53 |
| **6** | 洗碗机 | 32 |

### 2.7.2 代码实现

hive>

select t1.sku\_id,

name,

order\_num

from (

select sku\_id,

sum(sku\_num) order\_num

from order\_detail

where year(create\_date) = '2021'

and sku\_id in (

select sku\_id

from sku\_info

where datediff('2022-01-10', from\_date) > 30

)

group by sku\_id

having sum(sku\_num) < 100

) t1

left join

sku\_info t2

on t1.sku\_id = t2.sku\_id;

## 2.8 查询每日新用户数

### 2.8.1 题目需求

从用户登录明细表（user\_login\_detail）中查询每天的新增用户数，若一个用户在某天登录了，且在这一天之前没登录过，则任务该用户为这一天的新增用户。期望结果如下：

|  |  |
| --- | --- |
| **login\_date\_first（日期）** | **user\_count（新增用户数）** |
| **2021-09-21** | 1 |
| **2021-09-22** | 1 |
| **2021-09-23** | 1 |
| **2021-09-24** | 1 |
| **2021-09-25** | 1 |
| **2021-09-26** | 1 |
| **2021-09-27** | 1 |
| **2021-10-04** | 2 |
| **2021-10-06** | 1 |

### 2.8.2 代码实现

hive>

select

login\_date\_first,

count(\*) user\_count

from

(

select

user\_id,

min(date\_format(login\_ts,'yyyy-MM-dd')) login\_date\_first

from user\_login\_detail

group by user\_id

)t1

group by login\_date\_first;

## 2.9 统计每个商品的销量最高的日期

### 2.9.1 题目需求

从订单明细表（order\_detail）中统计出每种商品销售件数最多的日期及当日销量，如果有同一商品多日销量并列的情况，取其中的最小日期。期望结果如下：

|  |  |  |
| --- | --- | --- |
| **sku\_id（商品id）** | **create\_date（销量最高的日期）** | **sum\_num（销量）** |
| **1** | 2021-10-02 | 9 |
| **2** | 2021-10-04 | 60 |
| **3** | 2021-10-05 | 9 |
| **4** | 2021-10-07 | 10 |
| **5** | 2021-10-03 | 47 |
| **6** | 2021-10-03 | 8 |
| **7** | 2021-10-05 | 58 |
| **8** | 2021-10-08 | 59 |
| **9** | 2021-10-01 | 45 |
| **10** | 2021-10-08 | 94 |
| **11** | 2021-10-08 | 95 |
| **12** | 2021-10-08 | 83 |

### 2.9.2 代码实现

hive>

select sku\_id,

create\_date,

sum\_num

from (

select sku\_id,

create\_date,

sum\_num,

row\_number() over (partition by sku\_id order by sum\_num desc,create\_date asc) rn

from (

select sku\_id,

create\_date,

sum(sku\_num) sum\_num

from order\_detail

group by sku\_id, create\_date

) t1

) t2

where rn = 1;

## 2.10 查询销售件数高于品类平均数的商品

### 2.10.1 题目需求

从订单明细表（order\_detail）中查询累积销售件数高于其所属品类平均数的商品，期望结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **sku\_id** | **name** | **sum\_num** | **cate\_avg\_num** |
| **2** | 手机壳 | 302 | 110.5 |
| **5** | 破壁机 | 242 | 194.75 |
| **7** | 热水壶 | 252 | 194.75 |
| **8** | 微波炉 | 253 | 194.75 |
| **10** | 帐篷 | 299 | 290.5 |
| **11** | 烧烤架 | 320 | 290.5 |
| **12** | 遮阳伞 | 349 | 290.5 |

### 2.10.2 代码实现

hive>

select sku\_id,

name,

sum\_num,

cate\_avg\_num

from (

select od.sku\_id,

category\_id,

name,

sum\_num,

avg(sum\_num) over (partition by category\_id) cate\_avg\_num

from (

select sku\_id,

sum(sku\_num) sum\_num

from order\_detail

group by sku\_id

) od

left join

(

select sku\_id,

name,

category\_id

from sku\_info

) sku

on od.sku\_id = sku.sku\_id) t1

where sum\_num > cate\_avg\_num;

## 2.11 用户注册、登录、下单综合统计

### 2.11.1 题目需求

从用户登录明细表（user\_login\_detail）和订单信息表（order\_info）中查询每个用户的注册日期（首次登录日期）、总登录次数以及其在2021年的登录次数、订单数和订单总额。期望结果如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **user\_id**  **(用户id)** | **register\_date**  **(注册日期)** | **total\_login\_count**  **(累积登录次数)** | **login\_count\_2021**  **(2021年登录次数)** | **order\_count\_2021**  **(2021年下单次数)** | **order\_amount\_2021**  **(2021年订单金额)** |
| **101** | **2021-09-21** | **5** | **5** | **4** | **143660.00** |
| **102** | **2021-09-22** | **4** | **4** | **4** | **177850.00** |
| **103** | **2021-09-23** | **2** | **2** | **4** | **75890.00** |
| **104** | **2021-09-24** | **4** | **4** | **4** | **89880.00** |
| **105** | **2021-10-04** | **1** | **1** | **4** | **120100.00** |
| **106** | **2021-10-04** | **2** | **2** | **4** | **119150.00** |
| **107** | **2021-09-25** | **4** | **4** | **4** | **124150.00** |
| **108** | **2021-10-06** | **2** | **2** | **4** | **155770.00** |
| **109** | **2021-09-26** | **3** | **3** | **4** | **153500.00** |
| **1010** | **2021-09-27** | **2** | **2** | **4** | **51950.00** |

### 2.11.2 代码实现

hive>

select login.user\_id,

register\_date,

total\_login\_count,

login\_count\_2021,

order\_count\_2021,

order\_amount\_2021

from (

select user\_id,

min(date\_format(login\_ts, 'yyyy-MM-dd')) register\_date,

count(1) total\_login\_count,

count(if(year(login\_ts) = '2021', 1, null)) login\_count\_2021

from user\_login\_detail

group by user\_id

) login

join

(

select user\_id,

count(distinct(order\_id)) order\_count\_2021,

sum(total\_amount) order\_amount\_2021

from order\_info

where year(create\_date) = '2021'

group by user\_id

) oi

on login.user\_id = oi.user\_id;

## 2.12 查询指定日期的全部商品价格

### 2.12.1 题目需求

从商品价格修改明细表（sku\_price\_modify\_detail）中查询2021-10-01的全部商品的价格，假设所有商品初始价格默认都是99。期望结果如下：

|  |  |
| --- | --- |
| **sku\_id（商品id）** | **price（商品价格）** |
| **1** | **2000.00** |
| **2** | **10.00** |
| **3** | **5000.00** |
| **4** | **6000.00** |
| **5** | **500.00** |
| **6** | **2000.00** |
| **7** | **100.00** |
| **8** | **600.00** |
| **9** | **1000.00** |
| **10** | **90.00** |
| **11** | **66.00** |
| **12** | **20.00** |

### 2.12.2 代码实现

hive>

select sku\_info.sku\_id,

nvl(new\_price, 99) price

from sku\_info

left join

(

select sku\_id,

new\_price

from (

select sku\_id,

new\_price,

change\_date,

row\_number() over (partition by sku\_id order by change\_date desc) rn

from sku\_price\_modify\_detail

where change\_date <= '2021-10-01'

) t1

where rn = 1

) t2

on sku\_info.sku\_id = t2.sku\_id;

## 2.13 即时订单比例

### 2.13.1 题目需求

订单配送中，如果期望配送日期和下单日期相同，称为即时订单，如果期望配送日期和下单日期不同，称为计划订单。

请从配送信息表（delivery\_info）中求出每个用户的首单（用户的第一个订单）中即时订单的比例，保留两位小数，以小数形式显示。期望结果如下：

|  |
| --- |
| **percentage** |
| **0.5** |

### 2.13.2 代码实现

hive>

select

round(sum(if(order\_date=custom\_date,1,0))/count(\*),2) percentage

from

(

select

delivery\_id,

user\_id,

order\_date,

custom\_date,

row\_number() over (partition by user\_id order by order\_date) rn

from delivery\_info

)t1

where rn=1;

## 2.14 向用户推荐朋友收藏的商品

### 2.14.1 题目需求

现需要请向所有用户推荐其朋友收藏但是用户自己未收藏的商品，请从好友关系表（friendship\_info）和收藏表（favor\_info）中查询出应向哪位用户推荐哪些商品。期望结果如下：

**1）部分结果展示**

|  |  |
| --- | --- |
| **user\_id（用户id）** | **sku\_id（应向该用户推荐的商品id）** |
| **101** | **2** |
| **101** | **4** |
| **101** | **7** |
| **101** | **9** |
| **101** | **8** |
| **101** | **11** |
| **101** | **1** |

**2）完整结果**

user\_id sku\_id

101 2

101 4

101 7

101 9

101 8

101 11

101 1

102 3

102 5

102 10

103 2

103 1

103 9

104 1

104 4

104 10

104 5

104 2

105 1

105 2

105 6

105 12

105 3

106 11

106 10

106 8

106 9

106 3

107 11

107 7

107 4

107 9

107 12

107 1

107 8

107 6

107 2

108 2

108 6

108 12

108 1

108 7

108 4

108 5

109 6

109 10

109 7

109 1

109 12

109 3

109 11

1010 4

1010 10

1010 6

1010 12

1010 11

1010 8

1010 3

1010 5

1010 7

### 2.14.2 代码实现

hive>

select

distinct t1.user\_id,

friend\_favor.sku\_id

from

(

select

user1\_id user\_id,

user2\_id friend\_id

from friendship\_info

union

select

user2\_id,

user1\_id

from friendship\_info

)t1

left join favor\_info friend\_favor

on t1.friend\_id=friend\_favor.user\_id

left join favor\_info user\_favor

on t1.user\_id=user\_favor.user\_id

and friend\_favor.sku\_id=user\_favor.sku\_id

where user\_favor.sku\_id is null;

## 2.15 查询所有用户的连续登录两天及以上的日期区间

### 2.15.1 题目需求

从登录明细表（user\_login\_detail）中查询出，所有用户的连续登录两天及以上的日期区间，以登录时间（login\_ts）为准。期望结果如下：

|  |  |  |
| --- | --- | --- |
| **user\_id(用户id)** | **start\_date(开始日期)** | **end\_date(结束日期)** |
| **101** | 2021-09-27 | 2021-09-30 |
| **102** | 2021-10-01 | 2021-10-02 |
| **106** | 2021-10-04 | 2021-10-05 |
| **107** | 2021-10-05 | 2021-10-06 |

### 2.15.2 代码实现

hive>

select user\_id,

min(login\_date) start\_date,

max(login\_date) end\_date

from (

select user\_id,

login\_date,

date\_sub(login\_date, rn) flag

from (

select user\_id,

login\_date,

row\_number() over (partition by user\_id order by login\_date) rn

from (

select user\_id,

date\_format(login\_ts, 'yyyy-MM-dd') login\_date

from user\_login\_detail

group by user\_id, date\_format(login\_ts, 'yyyy-MM-dd')

) t1

) t2

) t3

group by user\_id, flag

having count(\*) >= 2;

## 2.16 男性和女性每日的购物总金额统计

### 2.16.1 题目需求

从订单信息表（order\_info）和用户信息表（user\_info）中，分别统计每天男性和女性用户的订单总金额，如果当天男性或者女性没有购物，则统计结果为0。期望结果如下：

|  |  |  |
| --- | --- | --- |
| **create\_date**  **（日期）** | **total\_amount\_male**  **（男性用户总金额）** | **total\_amount\_female**  **（女性用户总金额）** |
| **2021-09-27** | 29000.00 | 0.00 |
| **2021-09-28** | 70500.00 | 0.00 |
| **2021-09-29** | 43300.00 | 0.00 |
| **2021-09-30** | 860.00 | 0.00 |
| **2021-10-01** | 0.00 | 171680.00 |
| **2021-10-02** | 0.00 | 76150.00 |
| **2021-10-03** | 89880.00 | 5910.00 |
| **2021-10-04** | 9390.00 | 120100.00 |
| **2021-10-05** | 109760.00 | 69850.00 |
| **2021-10-06** | 101070.00 | 54300.00 |
| **2021-10-07** | 54700.00 | 129480.00 |
| **2021-10-08** | 51950.00 | 24020.00 |

### 2.16.2 代码实现

hive>

select create\_date,

sum(if(gender = '男', total\_amount, 0)) total\_amount\_male,

sum(if(gender = '女', total\_amount, 0)) total\_amount\_female

from order\_info oi

left join

user\_info ui

on oi.user\_id = ui.user\_id

group by create\_date;

## 2.17 订单金额趋势分析

### 2.17.1 题目需求

查询截止每天的最近3天内的订单金额总和以及订单金额日平均值，保留两位小数，四舍五入。期望结果如下：

|  |  |  |
| --- | --- | --- |
| **create\_date**  **（日期）** | **total\_3d**  **（最近3日订单金额总和）** | **avg\_ad**  **（最近3日订单金额日平均值）** |
| **2021-09-27** | 29000.00 | 29000.00 |
| **2021-09-28** | 99500.00 | 49750.00 |
| **2021-09-29** | 142800.00 | 47600.00 |
| **2021-09-30** | 114660.00 | 38220.00 |
| **2021-10-01** | 215840.00 | 71946.67 |
| **2021-10-02** | 248690.00 | 82896.67 |
| **2021-10-03** | 343620.00 | 114540.00 |
| **2021-10-04** | 301430.00 | 100476.67 |
| **2021-10-05** | 404890.00 | 134963.33 |
| **2021-10-06** | 464470.00 | 154823.33 |
| **2021-10-07** | 519160.00 | 173053.33 |
| **2021-10-08** | 415520.00 | 138506.67 |

### 2.17.2 代码实现

hive>

select create\_date,

round(sum(total\_amount\_by\_day) over (order by create\_date rows between 2 preceding and current row ),2) total\_3d,

round(avg(total\_amount\_by\_day) over (order by create\_date rows between 2 preceding and current row ), 2) avg\_3d

from (

select create\_date,

sum(total\_amount) total\_amount\_by\_day

from order\_info

group by create\_date

) t1;

## 2.18 购买过商品1和商品2但是没有购买商品3的顾客

### 2.18.1 题目需求

从订单明细表(order\_detail)中查询出所有购买过商品1和商品2，但是没有购买过商品3的用户，期望结果如下：

|  |
| --- |
| **user\_id** |
| **103** |
| **105** |

### 2.18.2 代码实现

hive>

select user\_id

from (

select user\_id,

collect\_set(sku\_id) skus

from order\_detail od

left join

order\_info oi

on od.order\_id = oi.order\_id

group by user\_id

) t1

where array\_contains(skus, '1')

and array\_contains(skus, '2')

and !array\_contains(skus, '3');

## 2.19 统计每日商品1和商品2销量的差值

### 2.19.1 题目需求

从订单明细表（order\_detail）中统计每天商品1和商品2销量（件数）的差值（商品1销量-商品2销量），期望结果如下：

|  |  |
| --- | --- |
| **create\_date** | **diff** |
| **2021-09-27** | 2 |
| **2021-10-01** | -10 |
| **2021-10-02** | -49 |
| **2021-10-03** | 4 |
| **2021-10-04** | -55 |
| **2021-10-05** | -30 |
| **2021-10-06** | -49 |
| **2021-10-07** | -40 |
| **2021-10-08** | -24 |

### 2.19.2 代码实现

hive>

select create\_date,

sum(if(sku\_id = '1', sku\_num, 0)) - sum(if(sku\_id = '2', sku\_num, 0)) diff

from order\_detail

where sku\_id in ('1', '2')

group by create\_date;

## 2.20 查询出每个用户的最近三笔订单

### 2.20.1 题目需求

从订单信息表（order\_info）中查询出每个用户的最近三笔订单，期望结果如下：

|  |  |  |
| --- | --- | --- |
| **user\_id** | **order\_id** | **create\_date** |
| **101** | 2 | 2021-09-28 |
| **101** | 3 | 2021-09-29 |
| **101** | 4 | 2021-09-30 |
| **102** | 5 | 2021-10-01 |
| **102** | 6 | 2021-10-01 |
| **102** | 8 | 2021-10-02 |
| **103** | 9 | 2021-10-02 |
| **103** | 10 | 2021-10-02 |
| **103** | 12 | 2021-10-03 |
| **104** | 13 | 2021-10-03 |
| **104** | 14 | 2021-10-03 |
| **104** | 15 | 2021-10-03 |
| **105** | 17 | 2021-10-04 |
| **105** | 18 | 2021-10-04 |
| **105** | 19 | 2021-10-04 |
| **106** | 22 | 2021-10-05 |
| **106** | 23 | 2021-10-05 |
| **106** | 24 | 2021-10-05 |
| **107** | 25 | 2021-10-05 |
| **107** | 27 | 2021-10-06 |
| **107** | 28 | 2021-10-06 |
| **108** | 29 | 2021-10-06 |
| **108** | 31 | 2021-10-07 |
| **108** | 32 | 2021-10-07 |
| **109** | 33 | 2021-10-07 |
| **109** | 35 | 2021-10-08 |
| **109** | 36 | 2021-10-08 |
| **1010** | 37 | 2021-10-08 |
| **1010** | 38 | 2021-10-08 |

### 2.20.2 代码实现

hive>

select user\_id,

order\_id,

create\_date

from (

select user\_id

, order\_id

, create\_date

, row\_number() over (partition by user\_id order by create\_date desc) rk

from order\_info

) t1

where rk <= 3;

## 2.21 查询每个用户登录日期的最大空档期

### 2.21.1 题目需求

从登录明细表（user\_login\_detail）中查询每个用户两个登录日期（以login\_ts为准）之间的最大的空档期。统计最大空档期时，用户最后一次登录至今的空档也要考虑在内，假设今天为2021-10-10。期望结果如下：

|  |  |
| --- | --- |
| **user\_id（用户id）** | **max\_diff（最大空档期）** |
| **101** | 10 |
| **102** | 9 |
| **103** | 10 |
| **104** | 9 |
| **105** | 6 |
| **106** | 5 |
| **107** | 10 |
| **108** | 4 |
| **109** | 10 |
| **1010** | 12 |

### 2.21.2 代码实现

hive>

select

user\_id,

max(diff) max\_diff

from

(

select

user\_id,

datediff(next\_login\_date,login\_date) diff

from

(

select

user\_id,

login\_date,

lead(login\_date,1,'2021-10-10') over(partition by user\_id order by login\_date) next\_login\_date

from

(

select

user\_id,

date\_format(login\_ts,'yyyy-MM-dd') login\_date

from user\_login\_detail

group by user\_id,date\_format(login\_ts,'yyyy-MM-dd')

)t1

)t2

)t3

group by user\_id;

## 2.22 查询相同时刻多地登陆的用户

### 2.22.1 题目需求

从登录明细表（user\_login\_detail）中查询在相同时刻，多地登陆（ip\_address不同）的用户，期望结果如下：

|  |
| --- |
| **user\_id(用户id)** |
| **101** |
| **102** |
| **104** |
| **107** |

### 2.22.2 代码实现

hive>

select

distinct t2.user\_id

from

(

select

t1.user\_id,

if(t1.max\_logout is null ,2,if(t1.max\_logout<t1.login\_ts,1,0)) flag

from

(

select

user\_id,

login\_ts,

logout\_ts,

max(logout\_ts)over(partition by user\_id order by login\_ts rows between unbounded preceding and 1 preceding) max\_logout

from

user\_login\_detail

)t1

)t2

where

t2.flag=0

## 2.23 销售额完成任务指标的商品

### 2.23.1 题目需求

商家要求每个商品每个月需要售卖出一定的销售总额

假设1号商品销售总额大于21000，2号商品销售总额大于10000，其余商品没有要求

请写出SQL从订单详情表中（order\_detail）查询连续两个月销售总额大于等于任务总额的商品

结果如下：

|  |
| --- |
| **sku\_id（商品id）** |
| **1** |

### 2.23.2 代码实现及步骤

hive>

**-- 求出1号商品 和 2号商品 每个月的购买总额 并过滤掉没有满足指标的商品**

select

sku\_id,

concat(substring(create\_date,0,7),'-01') ymd,

sum(price\*sku\_num) sku\_sum

from

order\_detail

where

sku\_id=1 or sku\_id=2

group by

sku\_id,substring(create\_date,0,7)

having

(sku\_id=1 and sku\_sum>=21000) or (sku\_id=2 and sku\_sum>=10000)

**-- 判断是否为连续两个月**

select

distinct t3.sku\_id

from

(

select

t2.sku\_id,

count(\*)over(partition by t2.sku\_id,t2.rymd) cn

from

(

select

t1.sku\_id,

add\_months(t1.ymd,-row\_number()over(partition by t1.sku\_id order by t1.ymd)) rymd

from

(

select

sku\_id,

concat(substring(create\_date,0,7),'-01') ymd,

sum(price\*sku\_num) sku\_sum

from

order\_detail

where

sku\_id=1 or sku\_id=2

group by

sku\_id,substring(create\_date,0,7)

having

(sku\_id=1 and sku\_sum>=21000) or (sku\_id=2 and sku\_sum>=10000)

)t1

)t2

)t3

where

t3.cn>=2

## 2.24 根据商品销售情况进行商品分类

### 2.24.1 题目需求

从订单详情表中（order\_detail）对销售件数对商品进行分类，0-5000为冷门商品，5001-19999位一般商品，20000往上为热门商品，并求出不同类别商品的数量

结果如下：

|  |  |
| --- | --- |
| **Category（类型）** | **Cn（数量）** |
| **一般商品** | **1** |
| **冷门商品** | **10** |
| **热门商品** | **1** |

### 2.24.2 代码实现

hive>

select

t2.category,

count(\*) cn

from

(

select

t1.sku\_id,

case

when t1.sku\_sum >=0 and t1.sku\_sum<=5000 then '冷门商品'

when t1.sku\_sum >=5001 and t1.sku\_sum<=19999 then '一般商品'

when t1.sku\_sum >=20000 then '热门商品'

end category

from

(

select

sku\_id,

sum(sku\_num) sku\_sum

from

order\_detail

group by

sku\_id

)t1

)t2

group by

t2.category

## 2.25 各品类销量前三的所有商品

### 2.25.1 题目需求

从订单详情表中（order\_detail）和商品（sku\_info）中查询各个品类销售数量前三的商品。如果该品类小于三个商品，则输出所有的商品销量。

结果如下：

|  |  |
| --- | --- |
| **Sku\_id（商品id）** | **Category\_id（品类id）** |
| **2** | 1 |
| **4** | 1 |
| **1** | 1 |
| **8** | 2 |
| **7** | 2 |
| **5** | 2 |
| **12** | 3 |
| **11** | 3 |
| **10** | 3 |

### 2.25.2 代码实现

hive>

select

t2.sku\_id,

t2.category\_id

from

(

select

t1.sku\_id,

si.category\_id,

rank()over(partition by category\_id order by t1.sku\_sum desc) rk

from

(

select

sku\_id,

sum(sku\_num) sku\_sum

from

order\_detail

group by

sku\_id

)t1

join

sku\_info si

on

t1.sku\_id=si.sku\_id

)t2

where

t2.rk<=3;

## 2.26 各品类中商品价格的中位数

### 2.26.1 题目需求

从商品（sku\_info）中球中位数如果是偶数则输出中间两个值的平均值，如果是奇数，则输出中间数即可。

结果如下：

|  |  |
| --- | --- |
| **Category\_id（品类id）** | **Medprice（中位数）** |
| **1** | 3500.0 |
| **2** | 1250.0 |
| **3** | 510.0 |

### 2.26.2 代码实现

hive>

**--求个每个品类 价格排序 商品数量 以及打上奇偶数的标签**

select

sku\_id,

category\_id,

price,

row\_number()over(partition by category\_id order by price desc) rk,

count(\*)over(partition by category\_id) cn,

count(\*)over(partition by category\_id)%2 falg

from

sku\_info t1

**--求出偶数品类的中位数**

select

distinct t1.category\_id,

avg(t1.price)over(partition by t1.category\_id) medprice

from

(

select

sku\_id,

category\_id,

price,

row\_number()over(partition by category\_id order by price desc) rk,

count(\*)over(partition by category\_id) cn,

count(\*)over(partition by category\_id)%2 falg

from

sku\_info

)t1

where

t1.falg=0 and (t1.rk=cn/2 or t1.rk=cn/2+1)

**--求出奇数品类的中位数**

select

t1.category\_id,

t1.price

from

(

select

sku\_id,

category\_id,

price,

row\_number()over(partition by category\_id order by price desc) rk,

count(\*)over(partition by category\_id) cn,

count(\*)over(partition by category\_id)%2 falg

from

sku\_info

)t1

where

t1.falg=1 and t1.rk=round(cn/2)

**-- 竖向拼接**

select

distinct t1.category\_id,

avg(t1.price)over(partition by t1.category\_id) medprice

from

(

select

sku\_id,

category\_id,

price,

row\_number()over(partition by category\_id order by price desc) rk,

count(\*)over(partition by category\_id) cn,

count(\*)over(partition by category\_id)%2 falg

from

sku\_info

)t1

where

t1.falg=0 and (t1.rk=cn/2 or t1.rk=cn/2+1)

union

select

t1.category\_id,

t1.price/1

from

(

select

sku\_id,

category\_id,

price,

row\_number()over(partition by category\_id order by price desc) rk,

count(\*)over(partition by category\_id) cn,

count(\*)over(partition by category\_id)%2 falg

from

sku\_info

)t1

where

t1.falg=1 and t1.rk=round(cn/2)

## 2.27 找出销售额连续3天超过100的商品

### 2.27.1 题目需求

从订单详情表（order\_detail）中找出销售额连续3天超过100的商品

结果如下：

|  |
| --- |
| **Sku\_id（商品id）** |
| **1** |
| **10** |
| **11** |
| **12** |
| **2** |
| **3** |
| **4** |
| **5** |
| **6** |
| **7** |
| **8** |
| **9** |

### 2.27.2 代码实现

hive>

**-- 每个商品每天的销售总额**

select

sku\_id,

create\_date,

sum(price\*sku\_num) sku\_sum

from

order\_detail

group by

sku\_id,create\_date

having

sku\_sum>=100

**-- 判断连续三天以上**

select

distinct t3.sku\_id

from

(

select

t2.sku\_id,

count(\*)over(partition by t2.sku\_id,t2.date\_drk) cdrk

from

(

select

t1.sku\_id,

t1.create\_date,

date\_sub(t1.create\_date,rank()over(partition by t1.sku\_id order by t1.create\_date)) date\_drk

from

(

select

sku\_id,

create\_date,

sum(price\*sku\_num) sku\_sum

from

order\_detail

group by

sku\_id,create\_date

having

sku\_sum>=100

)t1

)t2

)t3

where

t3.cdrk>=3

## 2.28 查询有新注册用户的当天的新用户数量、新用户的第一天留存率

### 2.28.1 题目需求

从用户登录明细表（user\_login\_detail）中首次登录算作当天新增，第二天也登录了算作一日留存

结果如下：

|  |  |  |
| --- | --- | --- |
| **first\_login（注册时间）** | **Register（新增用户数）** | **Retention（留存率）** |
| **2021-09-21** | 1 | 0.0 |
| **2021-09-22** | 1 | 0.0 |
| **2021-09-23** | 1 | 0.0 |
| **2021-09-24** | 1 | 0.0 |
| **2021-09-25** | 1 | 0.0 |
| **2021-09-26** | 1 | 0.0 |
| **2021-09-27** | 1 | 0.0 |
| **2021-10-04** | 2 | 0.5 |
| **2021-10-06** | 1 | 0.0 |

### 2.28.2 代码实现

hive>

**-- 每个用户首次登录时间 和 第二天是否登录 并看每天新增和留存数量**

select

t1.first\_login,

count(t1.user\_id) register,

count(t2.user\_id) remain\_1

from

(

select

user\_id,

date\_format(min(login\_ts),'yyyy-MM-dd') first\_login

from

user\_login\_detail

group by

user\_id

)t1

left join

user\_login\_detail t2

on

t1.user\_id=t2.user\_id and datediff(date\_format(t2.login\_ts,'yyyy-MM-dd'),t1.first\_login)=1

group by

t1.first\_login

**-- 新增数量和留存率**

select

t3.first\_login,

t3.register,

t3.remain\_1/t3.register retention

from

(

select

t1.first\_login,

count(t1.user\_id) register,

count(t2.user\_id) remain\_1

from

(

select

user\_id,

date\_format(min(login\_ts),'yyyy-MM-dd') first\_login

from

user\_login\_detail

group by

user\_id

)t1

left join

user\_login\_detail t2

on

t1.user\_id=t2.user\_id and datediff(date\_format(t2.login\_ts,'yyyy-MM-dd'),t1.first\_login)=1

group by

t1.first\_login

)t3

## 2.29 求出商品连续售卖的时间区间

### 2.29.1 题目需求

从订单详情表（order\_detail）中，求出商品连续售卖的时间区间

结果如下（截取部分）：

|  |  |  |
| --- | --- | --- |
| **Sku\_id（商品id）** | **Start\_date（起始时间）** | **End\_date（结束时间）** |
| **1** | 2021-09-27 | 2021-09-27 |
| **1** | 2021-09-30 | 2021-10-01 |
| **1** | 2021-10-03 | 2021-10-08 |
| **10** | 2021-10-02 | 2021-10-03 |
| **10** | 2021-10-05 | 2021-10-08 |
| **11** | 2021-10-02 | 2021-10-08 |
| **12** | 2021-09-30 | 2021-09-30 |
| **12** | 2021-10-02 | 2021-10-06 |
| **12** | 2021-10-08 | 2021-10-08 |

### 2.29.2 代码实现

hive>

**-- 每个商品售卖的日期以及拿到按排序后日期的差值**

select

sku\_id,

create\_date,

date\_sub(create\_date,rank()over(partition by sku\_id order by create\_date)) ddrk

from

order\_detail

group by

sku\_id,create\_date

**-- 拿到每次售卖的区间**

select

distinct

sku\_id,

first\_value(t1.create\_date)over(partition by t1.sku\_id,t1.ddrk order by t1.create\_date rows between unbounded preceding and unbounded following) start\_date,

last\_value(t1.create\_date)over(partition by t1.sku\_id,t1.ddrk order by t1.create\_date rows between unbounded preceding and unbounded following) end\_date

from

(

select

sku\_id,

create\_date,

date\_sub(create\_date,rank()over(partition by sku\_id order by create\_date)) ddrk

from

order\_detail

group by

sku\_id,create\_date

)t1

## 2.30 登录次数及交易次数统计

### 2.30.1 题目需求

分别从登陆明细表（user\_login\_detail）和配送信息表中用户登录时间和下单时间统计登陆次数和交易次数

结果如下（截取部分）：

|  |  |  |  |
| --- | --- | --- | --- |
| **User\_id**  **（用户id）** | **Login\_date**  **（登录时间）** | **login\_count**  **（登陆次数）** | **Order\_count**  **（交易次数）** |
| **101** | 2021-09-21 | 1 | 0 |
| **101** | 2021-09-27 | 1 | 1 |
| **101** | 2021-09-28 | 1 | 1 |
| **101** | 2021-09-29 | 1 | 1 |
| **101** | 2021-09-30 | 1 | 1 |
| **1010** | 2021-09-27 | 1 | 0 |
| **1010** | 2021-10-09 | 1 | 0 |
| **102** | 2021-09-22 | 1 | 0 |
| **102** | 2021-10-01 | 2 | 3 |

### 2.30.2 代码实现

hive>

**-- 拿到每个用户每天的登录次数**

select

user\_id,

date\_format(login\_ts,'yyyy-MM-dd') login\_date,

count(\*) login\_count

from

user\_login\_detail

group by

user\_id,date\_format(login\_ts,'yyyy-MM-dd')

**-- 拿到每个用户每天的交易次数**

select

t1.user\_id,

t1.login\_date,

collect\_set(t1.login\_count)[0] login\_count ,

count(di.user\_id) order\_count

from

(

select

user\_id,

date\_format(login\_ts,'yyyy-MM-dd') login\_date,

count(\*) login\_count

from

user\_login\_detail

group by

user\_id,date\_format(login\_ts,'yyyy-MM-dd')

)t1

left join

delivery\_info di

on

t1.user\_id=di.user\_id and t1.login\_date=di.order\_date

group by

t1.user\_id,t1.login\_date

## 2.31 按年度列出每个商品销售总额

### 2.31.1 题目需求

从订单明细表（order\_detail）中列出每个商品每个年度的购买总额

结果如下（截取部分）：

|  |  |  |
| --- | --- | --- |
| **Sku\_id（商品id）** | **Year\_date（年份）** | **Sku\_sum（销售总额）** |
| **1** | 2021 | 102000.00 |
| **10** | 2021 | 29900.00 |
| **11** | 2021 | 16000.00 |
| **12** | 2021 | 413640.00 |
| **2** | 2021 | 60440.00 |
| **3** | 2021 | 180000.00 |
| **4** | 2021 | 318000.00 |
| **5** | 2021 | 121000.00 |
| **6** | 2021 | 64000.00 |
| **7** | 2021 | 25200.00 |
| **8** | 2021 | 151800.00 |
| **9** | 2021 | 194000.00 |

### 2.31.2 代码实现

hive>

select

sku\_id,

year(create\_date) year\_date,

sum(price\*sku\_num) sku\_sum

from

order\_detail

group by

sku\_id,year(create\_date)

## 2.32. 某周内每件商品每天销售情况

### 2.32.1 题目需求

从订单详情表（order\_detail）中查询2021年9月27号-2021年10月3号这一周所有商品每天销售情况。

结果如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sku\_id**  **（商品id）** | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** | **Saturday** | **Sunday** |
| **1** | 0 | 0 | 9 | 8 | 0 | 4 | 2 |
| **10** | 0 | 0 | 0 | 0 | 48 | 69 | 0 |
| **11** | 0 | 0 | 0 | 0 | 15 | 61 | 0 |
| **12** | 0 | 0 | 43 | 0 | 31 | 20400 | 0 |
| **2** | 0 | 0 | 0 | 18 | 5800 | 0 | 0 |
| **3** | 0 | 0 | 0 | 6 | 0 | 1 | 5 |
| **4** | 9 | 0 | 0 | 8 | 1 | 5 | 0 |
| **5** | 33 | 0 | 0 | 0 | 24 | 47 | 0 |
| **6** | 0 | 0 | 0 | 1 | 5 | 8 | 0 |
| **7** | 0 | 37 | 0 | 17 | 0 | 20 | 0 |
| **8** | 0 | 46 | 0 | 48 | 39 | 0 | 0 |
| **9** | 0 | 12 | 0 | 45 | 0 | 0 | 0 |

### 2.32.2 代码实现

hive>

select

sku\_id,

sum(if(dayofweek(create\_date)=2,sku\_num,0)) Monday,

sum(if(dayofweek(create\_date)=3,sku\_num,0)) Tuesday,

sum(if(dayofweek(create\_date)=4,sku\_num,0)) Wednesday,

sum(if(dayofweek(create\_date)=5,sku\_num,0)) Thursday,

sum(if(dayofweek(create\_date)=6,sku\_num,0)) Friday,

sum(if(dayofweek(create\_date)=7,sku\_num,0)) Saturday,

sum(if(dayofweek(create\_date)=1,sku\_num,0)) Sunday

from

order\_detail

where

create\_date>='2021-09-27' and create\_date<='2021-10-03'

group by

sku\_id

## 2.33 查看每件商品的售价涨幅情况

### 2.33.1 题目需求

从商品价格变更明细表（sku\_price\_modify\_detail），得到最近一次价格的涨幅情况，并按照涨幅升序排序。

结果如下：

|  |  |
| --- | --- |
| **Sku\_id（商品id）** | **Price\_change（涨幅）** |
| **8** | -200.00 |
| **9** | -100.00 |
| **2** | -70.00 |
| **11** | -16.00 |
| **12** | -15.00 |
| **3** | 1.00 |
| **5** | 10.00 |
| **10** | 10.00 |
| **7** | 12.00 |
| **6** | 12.00 |
| **1** | 100.00 |
| **4** | 400.00 |

### 2.33.2 代码实现

hive>

**-- 对每个商品按照修改日期倒序排序 并求出差值**

select

sku\_id,

new\_price-lead(new\_price,1,0)over(partition by sku\_id order by change\_date desc) price\_change,

rank()over(partition by sku\_id order by change\_date desc) rk

from

sku\_price\_modify\_detail t1

**-- 最近一次修改的价格**

select

t1.sku\_id,

t1.price\_change

from

(

select

sku\_id,

new\_price-lead(new\_price,1,0)over(partition by sku\_id order by change\_date desc) price\_change,

rank()over(partition by sku\_id order by change\_date desc) rk

from

sku\_price\_modify\_detail

)t1

where

rk=1

order by

t1.price\_change

## 2.34 销售订单首购和次购分析

### 2.34.1 题目需求

通过商品信息表（sku\_info）订单信息表（order\_info）订单明细表（order\_detail）分析如果有一个用户成功下单两个及两个以上的购买成功的手机订单（购买商品为xiaomi 10，apple 12，小米13）那么输出这个用户的id及第一次成功购买手机的日期和第二次成功购买手机的日期，以及购买手机成功的次数。

结果如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **User\_id**  **（用户id）** | **First\_date**  **（首次时间）** | **Last\_value**  **（末次时间）** | **Cn**  **（购买次数）** |
| **101** | 2021-09-27 | 2021-09-28 | 3 |
| **1010** | 2021-10-08 | 2021-10-08 | 2 |
| **102** | 2021-10-01 | 2021-10-01 | 3 |
| **103** | 2021-09-30 | 2021-10-02 | 2 |
| **104** | 2021-10-03 | 2021-10-03 | 3 |
| **105** | 2021-10-04 | 2021-10-04 | 2 |
| **106** | 2021-10-04 | 2021-10-05 | 3 |
| **107** | 2021-10-05 | 2021-10-05 | 3 |
| **108** | 2021-10-06 | 2021-10-06 | 3 |
| **109** | 2021-10-07 | 2021-10-07 | 3 |

### 2.34.2 代码实现

hive>

select

distinct oi.user\_id,

first\_value(od.create\_date)over(partition by oi.user\_id order by od.create\_date rows between unbounded preceding and unbounded following ) first\_date,

last\_value(od.create\_date)over(partition by oi.user\_id order by od.create\_date rows between unbounded preceding and unbounded following ) last\_date,

count(\*)over(partition by oi.user\_id order by od.create\_date rows between unbounded preceding and unbounded following) cn

from

order\_info oi

join

order\_detail od

on

oi.order\_id=od.order\_id

join

sku\_info si

on

od.sku\_id=si.sku\_id

where

si.name in('xiaomi 10','apple 12','xiaomi 13')

## 2.35 同期商品售卖分析表

### 2.35.1 题目需求

从订单明细表（order\_detail）中。

求出同一个商品在2021年和2022年中同一个月的售卖情况对比。

结果如下（截取部分）：

|  |  |  |  |
| --- | --- | --- | --- |
| **Sku\_id**  **（商品id）** | **Month**  **（月份）** | **2020\_skusum**  **（2020销售量）** | **2021\_skusum**  **（2021销售量）** |
| **1** | 9 | 0 | 11 |
| **1** | 10 | 2 | 38 |
| **10** | 10 | 94 | 205 |
| **11** | 10 | 95 | 225 |
| **12** | 9 | 0 | 43 |
| **12** | 10 | 83 | 20556 |
| **2** | 10 | 26 | 6018 |
| **3** | 9 | 0 | 5 |
| **3** | 10 | 1 | 30 |
| **4** | 9 | 0 | 9 |

### 2.35.2 代码实现

hive>

select

if(t1.sku\_id is null,t2.sku\_id,t1.sku\_id),

month(if(t1.ym is null,t2.ym,t1.ym)) ,

if(t1.sku\_sum is null ,0 ,t1.sku\_sum) 2020\_skusum,

if(t2.sku\_sum is null ,0 ,t2.sku\_sum) 2020\_skusum

from

(

select

sku\_id,

concat(date\_format(create\_date,'yyyy-MM'),'-01') ym,

sum(sku\_num) sku\_sum

from

order\_detail

where

year(create\_date)=2020

group by

sku\_id,date\_format(create\_date,'yyyy-MM')

)t1

full join

(

select

sku\_id,

concat(date\_format(create\_date,'yyyy-MM'),'-01') ym,

sum(sku\_num) sku\_sum

from

order\_detail

where

year(create\_date)=2021

group by

sku\_id,date\_format(create\_date,'yyyy-MM')

)t2

on

t1.sku\_id=t2.sku\_id and month(t1.ym) = month(t2.ym)

## 2.36 国庆期间每个品类的商品的收藏量和购买量

### 2.36.1 题目需求

从订单明细表（order\_detail）和收藏信息表（favor\_info）统计2021国庆期间，每个商品总收藏量和购买量

结果如下：

|  |  |  |
| --- | --- | --- |
| **Sku\_id** | **Sku\_sum（购买量）** | **Favor\_cn（收藏量）** |
| **1** | 38 | 1 |
| **10** | 205 | 2 |
| **11** | 225 | 2 |
| **12** | 20556 | 0 |
| **2** | 6018 | 1 |
| **3** | 30 | 0 |
| **4** | 44 | 2 |
| **5** | 209 | 1 |
| **6** | 26 | 1 |
| **7** | 180 | 1 |
| **8** | 148 | 0 |
| **9** | 182 | 1 |

### 2.36.2 代码实现

hive>

select

t1.sku\_id,

t1.sku\_sum,

t2.favor\_cn

from

(

select

sku\_id,

sum(sku\_num) sku\_sum

from

order\_detail

where

create\_date>='2021-10-01' and create\_date<='2021-10-07'

group by

sku\_id

)t1

join

(

select

sku\_id,

count(\*) favor\_cn

from

favor\_info

where

create\_date>='2021-10-01' and create\_date<='2021-10-07'

group by

sku\_id

)t2

on

t1.sku\_id=t2.sku\_id

## 2.37 统计活跃间隔对用户分级结果

### 2.37.1 题目需求

用户等级：

忠实用户：近7天活跃且非新用户

新晋用户：近7天新增

沉睡用户：近7天未活跃但是在7天前活跃

流失用户：近30天未活跃但是在30天前活跃

假设今天是数据中所有日期的最大值，从用户登录明细表中的用户登录时间给各用户分级，求出各等级用户的人数

结果如下：

|  |  |
| --- | --- |
| **Level（用户等级）** | **Cn（用户数量)** |
| **忠实用户** | 6 |
| **新增用户** | 3 |
| **沉睡用户** | 1 |

### 2.37.2 代码实现

hive>

select

t2.level,

count(\*)

from

(

select

uld.user\_id,

case

when (date\_format(max(uld.login\_ts),'yyyy-MM-dd') <=date\_sub(today, 30))

then '流失用户'-- 最近登录时间三十天前

when (date\_format(min(uld.login\_ts),'yyyy-MM-dd') <=date\_sub(today, 7) and date\_format(max(uld.login\_ts),'yyyy-MM-dd') >=date\_sub(today, 7))

then '忠实用户' -- 最早登陆时间是七天前,并且最近七天登录过

when (date\_format(min(uld.login\_ts),'yyyy-MM-dd') >=date\_sub(today, 7))

then '新增用户' -- 最早登录时间是七天内

when (date\_format(min(uld.login\_ts),'yyyy-MM-dd') <= date\_sub(today, 7) and date\_format(max(uld.login\_ts),'yyyy-MM-dd') <= date\_sub(today, 7))

then '沉睡用户'-- 最早登陆时间是七天前,最大登录时间也是七天前

end level

from

user\_login\_detail uld

join

(

select

date\_format(max(login\_ts),'yyyy-MM-dd') today

from

user\_login\_detail

)t1

on

1=1

group by

uld.user\_id,t1.today

)t2

group by

t2.level

## 2.38 连续签到领金币数

### 2.38.1 题目需求

用户每天签到可以领1金币，并可以累计签到天数，连续签到的第3、7天分别可以额外领2和6金币。

每连续签到7天重新累积签到天数。

从用户登录明细表中求出每个用户金币总数，并按照金币总数倒序排序

结果如下：

|  |  |
| --- | --- |
| **User\_id（用户id）** | **Sum\_coin\_cn（金币总数)** |
| **101** | 7 |
| **109** | 3 |
| **107** | 3 |
| **102** | 3 |
| **106** | 2 |
| **104** | 2 |
| **103** | 2 |
| **1010** | 2 |
| **108** | 1 |
| **105** | 1 |

### 2.38.2 代码实现

hive>

**-- 求连续并标志是连续的第几天**

select

t1.user\_id,

t1.login\_date,

date\_sub(t1.login\_date,t1.rk) login\_date\_rk,

count(\*)over(partition by t1.user\_id, date\_sub(t1.login\_date,t1.rk) order by t1.login\_date) counti\_cn

from

(

select

user\_id,

date\_format(login\_ts,'yyyy-MM-dd') login\_date,

rank()over(partition by user\_id order by date\_format(login\_ts,'yyyy-MM-dd')) rk

from

user\_login\_detail

group by

user\_id,date\_format(login\_ts,'yyyy-MM-dd')

)t1

**--求出金币数量，以及签到奖励的金币数量**

select

t2.user\_id,

max(t2.counti\_cn)+sum(if(t2.counti\_cn%3=0,2,0))+sum(if(t2.counti\_cn%7=0,6,0)) coin\_cn

from

(

select

t1.user\_id,

t1.login\_date,

date\_sub(t1.login\_date,t1.rk) login\_date\_rk,

count(\*)over(partition by t1.user\_id, date\_sub(t1.login\_date,t1.rk) order by t1.login\_date) counti\_cn

from

(

select

user\_id,

date\_format(login\_ts,'yyyy-MM-dd') login\_date,

rank()over(partition by user\_id order by date\_format(login\_ts,'yyyy-MM-dd')) rk

from

user\_login\_detail

group by

user\_id,date\_format(login\_ts,'yyyy-MM-dd')

)t1

)t2

group by

t2.user\_id,t2.login\_date\_rk

**-- 求出每个用户的金币总数**

select

t3.user\_id,

sum(t3.coin\_cn) sum\_coin\_cn

from

(

select

t2.user\_id,

max(t2.counti\_cn)+sum(if(t2.counti\_cn%3=0,2,0))+sum(if(t2.counti\_cn%7=0,6,0)) coin\_cn

from

(

select

t1.user\_id,

t1.login\_date,

date\_sub(t1.login\_date,t1.rk) login\_date\_rk,

count(\*)over(partition by t1.user\_id, date\_sub(t1.login\_date,t1.rk) order by t1.login\_date) counti\_cn

from

(

select

user\_id,

date\_format(login\_ts,'yyyy-MM-dd') login\_date,

rank()over(partition by user\_id order by date\_format(login\_ts,'yyyy-MM-dd')) rk

from

user\_login\_detail

group by

user\_id,date\_format(login\_ts,'yyyy-MM-dd')

)t1

)t2

group by

t2.user\_id,t2.login\_date\_rk

)t3

group by

t3.user\_id

order by

sum\_coin\_cn desc

## 2.39 国庆期间的7日动销率和滞销率

### 2.39.1 题目需求

动销率定义为品类商品中一段时间内有销量的商品占当前已上架总商品数的比例（有销量的商品/已上架总商品数）。

滞销率定义为品类商品中一段时间内没有销量的商品占当前已上架总商品数的比例。（没有销量的商品 / 已上架总商品数）。

只要当天任一店铺有任何商品的销量就输出该天的结果

从订单明细表（order\_detail）和商品信息表（sku\_info）表中求出国庆7天每天每个品类的商品的动销率和滞销率

结果如下（截取部分）：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Category\_id**  **（品类id）** | **1号**  **（动销)** | **1号**  **（滞销)** | **2号**  **（动销)** | **2号**  **（滞销)** | **3号**  **（动销)** | **3号**  **（滞销)** |
| **1** | 1.0 | 0.0 | 0.5 | 0.5 | 0.75 | 0.25 |
| **2** | 0.75 | 0.25 | 0.75 | 0.25 | 0.75 | 0.25 |
| **3** | 0.25 | 0.75 | 0.75 | 0.25 | 0.75 | 0.25 |

### 2.39.2 代码实现

hive>

**-- 国庆每一天 每个商品品类有多少商品被销售了**

select

t1.category\_id,

sum(if(t1.create\_date='2021-10-01',1,0)) `第1天`,

sum(if(t1.create\_date='2021-10-02',1,0)) `第2天`,

sum(if(t1.create\_date='2021-10-03',1,0)) `第3天`,

sum(if(t1.create\_date='2021-10-04',1,0)) `第4天`,

sum(if(t1.create\_date='2021-10-05',1,0)) `第5天`,

sum(if(t1.create\_date='2021-10-06',1,0)) `第6天`,

sum(if(t1.create\_date='2021-10-07',1,0)) `第7天`

from

(

select

distinct

si.category\_id,

od.create\_date,

si.name

from

order\_detail od

join

sku\_info si

on

od.sku\_id=si.sku\_id

where

od.create\_date>='2021-10-01' and od.create\_date<='2021-10-07'

)t1

group by

t1.category\_id

**-- 每一天的动销率 和 滞销率**

select

t2.category\_id,

t2.`第1天`/t3.cn,

1-t2.`第1天`/t3.cn,

t2.`第2天`/t3.cn,

1-t2.`第2天`/t3.cn,

t2.`第3天`/t3.cn,

1-t2.`第3天`/t3.cn,

t2.`第4天`/t3.cn,

1-t2.`第4天`/t3.cn,

t2.`第5天`/t3.cn,

1-t2.`第5天`/t3.cn,

t2.`第6天`/t3.cn,

1-t2.`第6天`/t3.cn,

t2.`第7天`/t3.cn,

1-t2.`第7天`/t3.cn

from

(

select

t1.category\_id,

sum(if(t1.create\_date='2021-10-01',1,0)) `第1天`,

sum(if(t1.create\_date='2021-10-02',1,0)) `第2天`,

sum(if(t1.create\_date='2021-10-03',1,0)) `第3天`,

sum(if(t1.create\_date='2021-10-04',1,0)) `第4天`,

sum(if(t1.create\_date='2021-10-05',1,0)) `第5天`,

sum(if(t1.create\_date='2021-10-06',1,0)) `第6天`,

sum(if(t1.create\_date='2021-10-07',1,0)) `第7天`

from

(

select

distinct

si.category\_id,

od.create\_date,

si.name

from

order\_detail od

join

sku\_info si

on

od.sku\_id=si.sku\_id

where

od.create\_date>='2021-10-01' and od.create\_date<='2021-10-07'

)t1

group by

t1.category\_id

)t2

join

(

select

category\_id,

count(\*) cn

from

sku\_info

group by

category\_id

)t3

on

t2.category\_id=t3.category\_id

## 2.40 同时在线最多的人数

### 2.40.1 题目需求

根据用户登录明细表（user\_login\_detail），求出平台同时在线最多的人数。

结果如下：

|  |
| --- |
| **Cn（人数）** |
| **7** |

### 2.40.2 代码实现

hive>

**-- 登录标记1 下线标记-1**

select

login\_ts l\_time,

1 flag

from

user\_login\_detail

union

select

logout\_ts l\_time,

-1 flag

from

user\_login\_detail

**-- 按照时间求和**

select

sum(flag)over(order by t1.l\_time) sum\_l\_time

from

(

select

login\_ts l\_time,

1 flag

from

user\_login\_detail

union

select

logout\_ts l\_time,

-1 flag

from

user\_login\_detail

)t1

**-- 拿到最大值 就是同时在线最多人数**

select

max(sum\_l\_time)

from

(

select

sum(flag)over(order by t1.l\_time) sum\_l\_time

from

(

select

login\_ts l\_time,

1 flag

from

user\_login\_detail

union

select

logout\_ts l\_time,

-1 flag

from

user\_login\_detail

)t1

)t2