1. write a query to find number of gold medal per swimmer for swimmers who won only gold medals

```
select gold as player_name, Count(1) as no_of_medals
from events
where gold not in (select silver from events union all select bronze from events)
group by gold
```

2. Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.

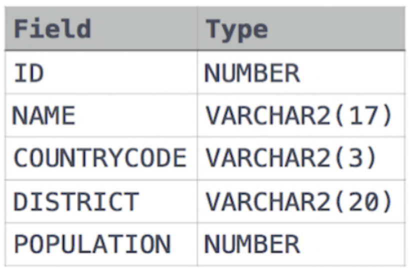
```
SELECT *
FROM CITY
WHERE CountryCode = 'USA' AND Population > 100000;
```

3. Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.

```
SELECT NAME
FROM city
WHERE Population > 120000 AND CountryCode = 'USA';
```

4. Query all columns (attributes) for every row in the CITY table.

CITY



Field	Type
ID	NUMBER
NAME	VARCHAR2(17)
COUNTRYCODE	VARCHAR2(3)
DISTRICT	VARCHAR2(20)
POPULATION	NUMBER

```
select * FROM city ;
```

5. Query all attributes of every Japanese city in the CITY table. The COUNTRYCODE for Japan is JPN.

```
select * from city where COUNTRYCODE = 'JPN' ;
```

6. Query the names of all the Japanese cities in the CITY table. The COUNTRYCODE for Japan is JPN.

select name from city where COUNTRYCODE = 'JPN' ;

7. **Query a list of CITY and STATE from the STATION table.**

SELECT CITY, STATE FROM STATION ;

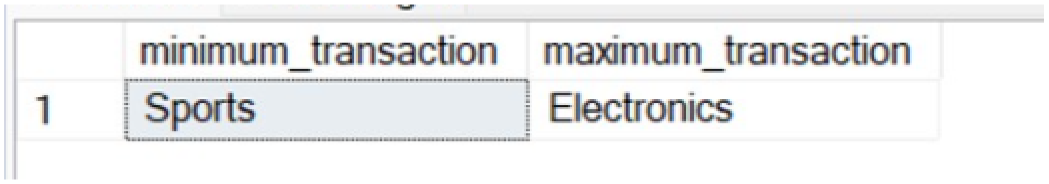
29-02-2024

8. **Problem Statement:**

Write an SQL query to display the dept which has the overall minimum txnamount and the dept which has the overall maximum transaction amount.

Table Script:
create table ecommerce(dept varchar(50),txnmonth varchar(50),txnamount int);
insert into ecommerce values('Electronics','Jan',1000);
insert into ecommerce values('Electronics','Feb',2000);
insert into ecommerce values('Electronics','Mar',2500);
insert into ecommerce values('Textile','Jan',1000);
insert into ecommerce values('Textile','Feb',2000);
insert into ecommerce values('Textile','Mar',1500);
insert into ecommerce values('Sports','Jan',600);
insert into ecommerce values('Sports','Feb',200);
insert into ecommerce values('Sports','Mar',500);

SOLUTION :



	minimum_transaction	maximum_transaction
1	Sports	Electronics

Solution -1

select * from ecommerce ;
select
(select dept from ecommerce where txnamount = (select min(txnamount) from ecommerce)) as a
,
(select dept from ecommerce where txnamount = (select max(txnamount) from ecommerce)) as b;

Solution -2

SELECT
(SELECT dept FROM ecommerce WHERE txnamount = (SELECT MIN(txnamount) FROM ecommerce)) AS min_txnamt_dept,

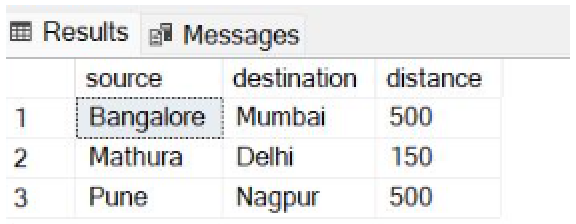
(SELECT dept FROM ecommerce WHERE txnamount = (SELECT MAX(txnamount) FROM ecommerce)) AS max_txnamt_dept;

9. **Problem Statement**:

Given the travel table with columns **source**, **destination**, **and distance**. Write an SQL query **to find all unique travel routes**.
Let's consider we have data where travel routes exist between Bangalore to Mumbai, and also between Mumbai to Bangalore. For such data, only one route should be considered.

Table Script:
CREATE TABLE travel (source VARCHAR(512),
destination VARCHAR(512),
distance INT
);
INSERT INTO travel VALUES ('Mumbai', 'Bangalore', '500'),
('Bangalore', 'Mumbai', '500'),
('Delhi', 'Mathura', '150'),
('Mathura', 'Delhi', '150'),
('Nagpur', 'Pune', '500'),
('Pune', 'Nagpur', '500');

SOLUTION -



	source	destination	distance
1	Bangalore	Mumbai	500
2	Mathura	Delhi	150
3	Pune	Nagpur	500

SOLUTION -1

select * from travel where source < destination ;

SOLUTION - 2

SELECT DISTINCT t1.source, t1.destination, t1.distance
FROM travel t1, travel t2
WHERE t1.source = t2.destination
AND t1.destination = t2.source
AND t1.source < t1.destination;

SOLUTION-3

SELECT DISTINCT *
FROM travel

```
WHERE source > destination;
```

10. Problem Statement:

Write an SQL query to find the gender who has done highest transaction amount per month per city.

Table Script:

```
CREATE TABLE customer (
customer_id INT,
cust_name VARCHAR(255),
city VARCHAR(255),
gender CHAR(1)
);
INSERT INTO customer VALUES (3006, 'Geoff Cameron', 'London', 'M');
INSERT INTO customer VALUES (3005, 'Graham Zusi', 'London', 'F');
INSERT INTO customer VALUES (3004, 'Julian Green', 'London', 'M');
INSERT INTO customer VALUES (3003, 'Jozy Altidor', 'Bangalore', 'F');
INSERT INTO customer VALUES (3002, 'Nick Rimando', 'Bangalore', 'M');
INSERT INTO customer VALUES (3001, 'Brad Guzan', 'Bangalore', 'F');

CREATE TABLE orderdetails (
ord_no INT,
purch_amt DECIMAL(10, 2),
ord_date DATE,
customer_id INT
);
INSERT INTO orderdetails VALUES (70009, 2700, '2012-09-10', 3001);
INSERT INTO orderdetails VALUES (70008, 1760, '2012-09-10', 3002);
INSERT INTO orderdetails VALUES (70002, 500, '2012-10-05', 3002);
INSERT INTO orderdetails VALUES (70002, 250, '2012-10-05', 3003);
INSERT INTO orderdetails VALUES (70013, 3045, '2012-09-25', 3004);
INSERT INTO orderdetails VALUES (70013, 685, '2012-09-25', 3005);
INSERT INTO orderdetails VALUES (70011, 752, '2012-10-17', 3005);
INSERT INTO orderdetails VALUES (70010, 1983, '2012-10-10', 3006);
```

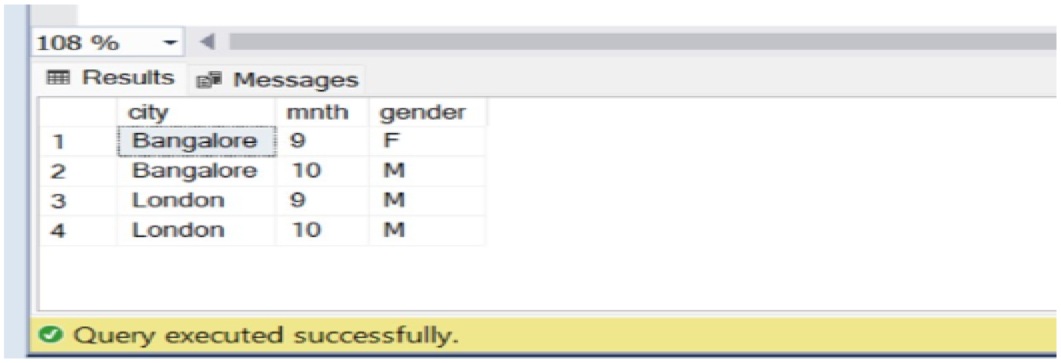
SOLUTION1. -

```
SELECT
   b.city,
   MONTH(a.ord_date) AS month,
   b.gender
FROM
   orderdetails AS a
```

```
INNER JOIN
   customer AS b ON a.customer_id = b.customer_id
WHERE
   a.purch_amt = (
      SELECT
         MAX(a1.purch_amt)
      FROM
         orderdetails AS a1
      INNER JOIN
         customer AS b1 ON a1.customer_id = b1.customer_id
      WHERE
         MONTH(a1.ord_date) = MONTH(a.ord_date)
         AND b1.city = b.city
   )
GROUP BY
   b.city, MONTH(a.ord_date), b.gender;
```

Solution-2

```
WITH cte1
AS (
SELECT c.city, DATEPART(MONTH, o.ord_date) AS mnth, c.gender, SUM(o.purch_amt) AS
total_amount
FROM customer c
INNER JOIN orderdetails o ON c.customer_id = o.customer_id
GROUP BY c.city, DATEPART(MONTH, o.ord_date), c.gender
)
SELECT a.city, a.mnth, a.gender
FROM (
SELECT *
,DENSE_RANK() OVER (PARTITION BY city, mnth ORDER BY total_amount DESC) AS
drnk
FROM cte1
) a
WHERE a.drnk = 1;
```


108 %

Results | Messages

	city	mnth	gender
1	Bangalore	9	F
2	Bangalore	10	M
3	London	9	M
4	London	10	M

Query executed successfully.

11. Write an SQL query to fetch the user_ids which have only bought 'Burger' and 'Cold Drink' and no other items.

Script:

```
CREATE TABLE orders(
user_id INT,
item_ordered VARCHAR(512)
);
INSERT INTO orders VALUES ('1', 'Pizza');
INSERT INTO orders VALUES ('1', 'Burger');
INSERT INTO orders VALUES ('2', 'Cold Drink');
INSERT INTO orders VALUES ('2', 'Burger');
INSERT INTO orders VALUES ('3', 'Burger');
INSERT INTO orders VALUES ('3', 'Cold Drink');
INSERT INTO orders VALUES ('4', 'Pizza');
INSERT INTO orders VALUES ('4', 'Cold Drink');
INSERT INTO orders VALUES ('5', 'Cold Drink');
INSERT INTO orders VALUES ('6', 'Burger');
INSERT INTO orders VALUES ('6', 'Cold Drink');
INSERT INTO orders VALUES ('7', 'Pizza');
INSERT INTO orders VALUES ('8', 'Burger');
```

ANSWER :

Solution 1

```
select distinct a.user_id from orders as a
where a.user_id in (
select user_id from orders where item_ordered = "Burger"
        intersect
        select user_id from orders where item_ordered = "Cold Drink"
        )
        and
        a.user_id not in (
select user_id from orders where item_ordered = "Pizza" ) ;
```

Solution 2

```
SELECT user_id
FROM orders
GROUP BY user_id
HAVING COUNT(DISTINCT item_ordered) = 2
AND SUM(CASE WHEN item_ordered IN ('Burger', 'Cold Drink') THEN 1 ELSE 0 END)
= 2;
```

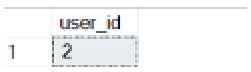
1/03/2024

12. Write an SQL query to find the users who exclusively purchased iPhone 15 only and did not buy any other iPhone model.

Table Script:

```
CREATE TABLE iphone_dataset
(
user_id VARCHAR(20),
iphone_model VARCHAR(20)
);
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('1', 'i-11');
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('1', 'i-12');
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('1', 'i-13');
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('1', 'i-14');
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('1', 'i-15');
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('2', 'i-15');
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('3', 'i-12');
INSERT INTO iphone_dataset (user_id, iphone_model) VALUES ('3', 'i-15');

CREATE TABLE iphone_products_dim
(
iphone_model VARCHAR(20)
);
INSERT INTO iphone_products_dim (iphone_model) VALUES ('i-11');
INSERT INTO iphone_products_dim (iphone_model) VALUES ('i-12');
INSERT INTO iphone_products_dim (iphone_model) VALUES ('i-13');
INSERT INTO iphone_products_dim (iphone_model) VALUES ('i-14');
INSERT INTO iphone_products_dim (iphone_model) VALUES ('i-15');
```

Solution 1- 

	user_id
1	2

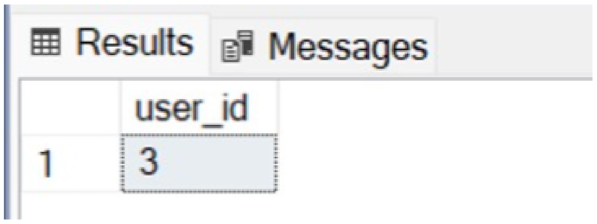
```
SELECT user_id from iphone_dataset
WHERE iphone_model = 'i-15'
AND user_id NOT IN (
    SELECT user_id FROM iphone_dataset
    WHERE iphone_model IN ('i-11', 'i-12', 'i-13', 'i-14')
);

select user_id from iphone_dataset
where iphone_model = 'i-15'
and user_id not in(
select user_id from iphone_dataset
where iphone_model in('i-11', 'i-12', 'i-13', 'i-14')
);
```

Solution - 2

```
select user_id from
(select user_id from iphone_dataset where  iphone_model = "i-15"
except
select user_id from
(select user_id from iphone_dataset where  iphone_model = "i-11"
union
select user_id from iphone_dataset where  iphone_model = "i-12"
union
select user_id from iphone_dataset where  iphone_model = "i-13"
union
select user_id from iphone_dataset where  iphone_model = "i-14") as a
) as b
```

12 (b) Write an SQL query to identify users who have upgraded their iPhone model from iPhone 12 to iPhone 15, and they have only purchased two iPhone models in total.



	user_id
1	3

Solution-1

```
select user_id from
(select user_id from iphone_dataset where
iphone_model = "i-12"
intersect
select user_id from iphone_dataset where
iphone_model = "i-15" ) as a
intersect
select user_id from iphone_dataset group by user_id having
count(iphone_model) = 2;
```

12) c) Consider you are having iphone_products_dim table which contains all the available iphone models. Write an SQL query to find the users who have purchased every iPhone model listed in the iphone_products_dim table.

<u>Solution - 1</u>

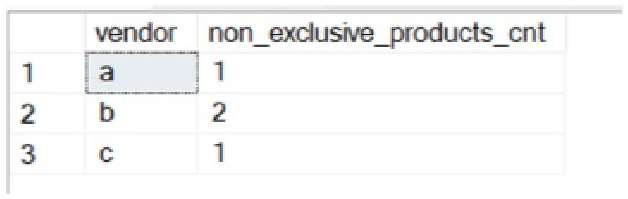
```
select user_id from
(
select user_id from iphone_dataset where iphone_model = "i-11"
intersect
select user_id from iphone_dataset where iphone_model = "i-12"
intersect
select user_id from iphone_dataset where iphone_model = "i-13"
intersect
select user_id from iphone_dataset where iphone_model = "i-14"
intersect
select user_id from iphone_dataset where iphone_model = "i-15"
) as a;
```

13) Write an SQL query to find out the count of Non exclusive products for a vendor.
Non exclusive products: A product which is not available in any other vendors in the table.

Table Script:

```
create table tbl (vendor varchar(10), product varchar(20));
insert into tbl values ('a','grape');
insert into tbl values ('a','peach');
insert into tbl values ('a','orange');
insert into tbl values ('b','peach');
insert into tbl values ('b','strawberry');
insert into tbl values ('b','apple');
insert into tbl values ('c','grape');
insert into tbl values ('c','peach');
insert into tbl values ('c','pineapple');
```

Solution 1-



	vendor	non_exclusive_products_cnt
1	a	1
2	b	2
3	c	1

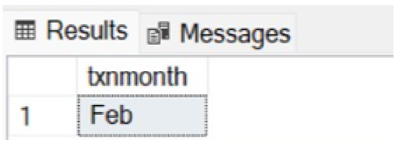
```
select b.vendor, count(b.product) as units from tbl as b
inner join
(select product,count(vendor)  from tbl group by product having count(vendor) =
1 ) as a
on b.product = a.product
group by vendor;
```

<u>2/03/2024</u>

1. **Write an SQL query to find the txnmonth which has the maximum txnamount.**

```
create table eshop(txnmonth varchar(50),clothing int,electronics int,sports int);
insert into eshop values('Jan',2000,1500,3000);
insert into eshop values('Feb',1000,2500,4000);
insert into eshop values('Mar',2000,1400,1000);
insert into eshop values('Apr',3000,1500,1000);
```

Solution -



	txnmonth
1	Feb

```
WITH max_txn_amount AS (
   SELECT txnmonth, SUM(clothing + electronics + sports) AS total_txn_amount
   FROM eshop
   GROUP BY txnmonth
)
SELECT m.txnmonth
FROM max_txn_amount m
WHERE m.total_txn_amount = (SELECT MAX(total_txn_amount) FROM
max_txn_amount);
```

Solution-2

```
SELECT txnmonth
FROM eshop
GROUP BY txnmonth
```

```
ORDER BY SUM(clothing + electronics + sports) DESC
LIMIT 1;
```

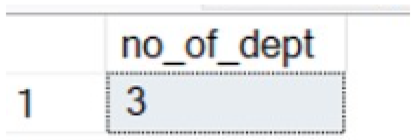
Solution-3

```
select txnmonth from eshop
where
clothing+ electronics+ sports
= (select max(clothing+ electronics+ sports) as max_txn from eshop )
```

2. Write an SQL query to find the count of distinct departments including NULL.

Table Script:

```
create table department(deptid int, deptname varchar(50));
insert into department values(1,'Tech');
insert into department values(2,'HR');
insert into department values(3,null);
insert into department values(4,'Tech');
insert into department values(5,'HR');
```



	no_of_dept
1	3

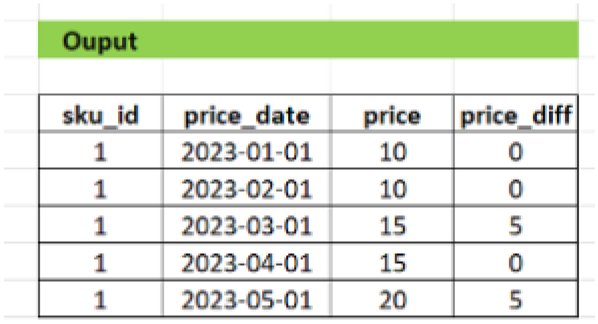
SOLUTION -

```
select count(*) as count_dist_dept from
(select distinct deptname from department group by deptname) as a;
```

2. You are given the price of each sku whenever there is a change in price. Write an SQL query to find the price at the start of each month & calculate the difference from the previous month's start date.

Table Script:

```
create table sku (sku_id int,
price_date date ,
price int
);
insert into sku values (1,'2023-01-01',10)
,(1,'2023-02-15',15)
,(1,'2023-03-03',18)
,(1,'2023-03-27',15)
,(1,'2023-04-06',20);
```



Ouput

sku_id	price_date	price	price_diff
1	2023-01-01	10	0
1	2023-02-01	10	0
1	2023-03-01	15	5
1	2023-04-01	15	0
1	2023-05-01	20	5

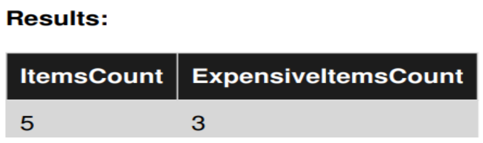
Solution -

3/03/2024

1. **Write an SQL query to find the total number of items that have been categorized as "Expensive".**

Table script :

```
CREATE TABLE product_price
(Id int, ItemId int, Price decimal(10, 2), PriceRating varchar(10)) ;
INSERT INTO product_price
(Id, ItemId, Price, PriceRating)
VALUES
(1, 100, 34.5, "EXPENSIVE") ,
(2, 145, 2.3, "CHEAP") ,
(3, 100, 34.5, "EXPENSIVE") ,
(4, 100, 34.5, "EXPENSIVE") ,
(5, 145, 10, "AFFORDABLE") ;
```



Results:

ItemsCount	ExpensiveItemsCount
5	3

SOLUTION 1-

```
select
(select count(*) from product_price) as totalitemscount,
(select count(*) from product_price where PriceRating = 'EXPENSIVE') as
totalexpensiveitems;
```

Solution - 2

```
select
count(id) as itemscount,
sum(case when PriceRating = 'EXPENSIVE' then 1 else 0 end) as
expensiveitemscount
```

from
Product_price;

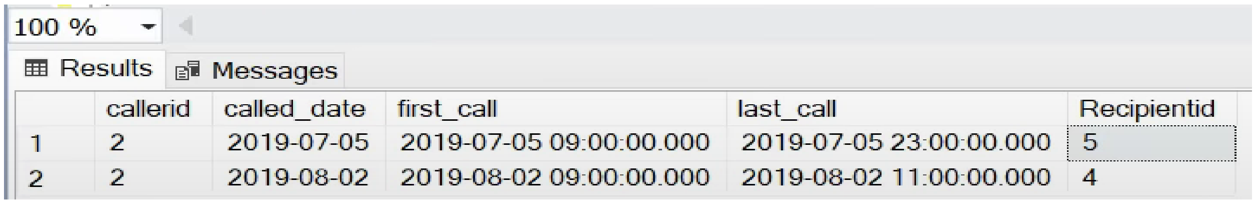
2. **There is a phone log table that has information about callers call history. Write a sql to find out callers whose first and last call was to the same person on a given day.**

Table script:

```
create table phonelog(
   Callerid int,
   Recipientid int,
   Datecalled datetime
);

insert into phonelog(Callerid, Recipientid, Datecalled)
values(1, 2, '2019-01-01 09:00:00.000'),
      (1, 3, '2019-01-01 17:00:00.000'),
      (1, 4, '2019-01-01 23:00:00.000'),
      (2, 5, '2019-07-05 09:00:00.000'),
      (2, 3, '2019-07-05 17:00:00.000'),
      (2, 3, '2019-07-05 17:20:00.000'),
      (2, 5, '2019-07-05 23:00:00.000'),
      (2, 3, '2019-08-01 09:00:00.000'),
      (2, 3, '2019-08-01 17:00:00.000'),
      (2, 5, '2019-08-01 19:30:00.000'),
      (2, 4, '2019-08-02 09:00:00.000'),
      (2, 5, '2019-08-02 10:00:00.000'),
      (2, 5, '2019-08-02 10:45:00.000'),
      (2, 4, '2019-08-02 11:00:00.000');
```

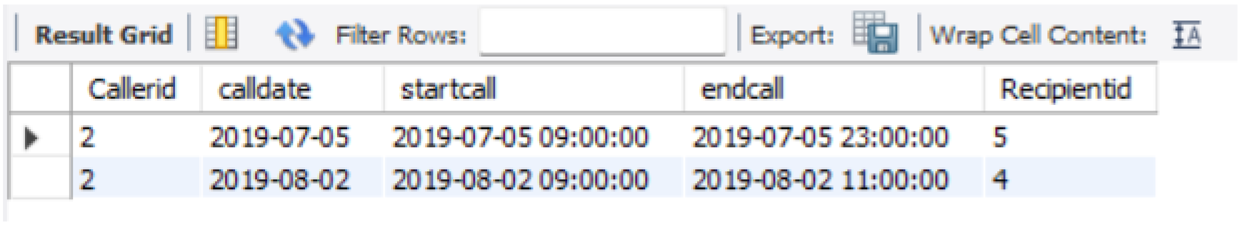
Solution -



	callerid	called_date	first_call	last_call	Recipientid
1	2	2019-07-05	2019-07-05 09:00:00.000	2019-07-05 23:00:00.000	5
2	2	2019-08-02	2019-08-02 09:00:00.000	2019-08-02 11:00:00.000	4

```
select a.Callerid, b.calldate, b.startcall, b.endcall, a.Recipientid from phonelog as a
inner join
(select date(Datecalled) as calldate, min(Datecalled) as startcall, max(Datecalled) as endcall
 from phonelog group by calldate) as b
 on a.Datecalled = b.startcall
 intersect
```

```
select c.Callerid, d.calldate, d.startcall, d.endcall, c.Recipientid from phonelog as c
inner join
(select date(Datecalled) as calldate, min(Datecalled) as startcall, max(Datecalled) as endcall
 from phonelog group by calldate) as d
 on c.Datecalled = d.endcall ;
```

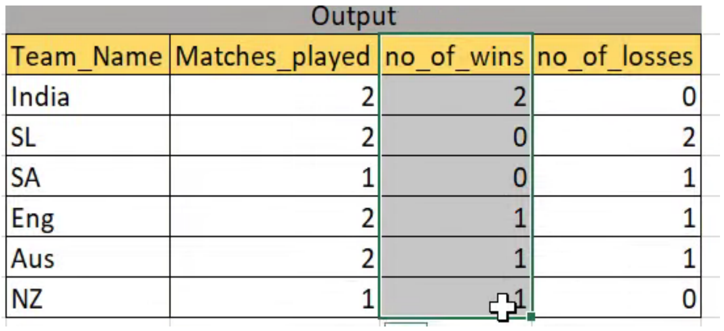


Callerid	calldate	startcall	endcall	Recipientid
2	2019-07-05	2019-07-05 09:00:00	2019-07-05 23:00:00	5
2	2019-08-02	2019-08-02 09:00:00	2019-08-02 11:00:00	4

3. Derive Points table for ICC tournament

```
create table icc_world_cup
(
Team_1 Varchar(20),
Team_2 Varchar(20),
Winner Varchar(20)
);
INSERT INTO icc_world_cup values('India','SL','India');
INSERT INTO icc_world_cup values('SL','Aus','Aus');
INSERT INTO icc_world_cup values('SA','Eng','Eng');
INSERT INTO icc_world_cup values('Eng','NZ','NZ');
INSERT INTO icc_world_cup values('Aus','India','India');
```

Solution



Output

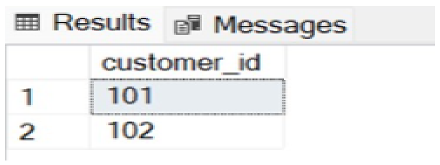
Team_Name	Matches_played	no_of_wins	no_of_losses
India	2	2	0
SL	2	0	2
SA	1	0	1
Eng	2	1	1
Aus	2	1	1
NZ	1	1	0

4. **Write an SQL query to find the list of customer id's who have purchased for 3 or more consecutive days.**

```
CREATE TABLE purchases (purchase_id INT,
customer_id INT,
purchase_date DATE
);
```

```
INSERT INTO purchases VALUES (1, 101, '2024-01-01');
INSERT INTO purchases VALUES (2, 102, '2024-01-02');
INSERT INTO purchases VALUES (3, 101, '2024-01-02');
INSERT INTO purchases VALUES (4, 103, '2024-01-03');
INSERT INTO purchases VALUES (5, 101, '2024-01-03');
INSERT INTO purchases VALUES (6, 104, '2024-01-04');
INSERT INTO purchases VALUES (7, 102, '2024-01-04');
INSERT INTO purchases VALUES (8, 103, '2024-01-05');
INSERT INTO purchases VALUES (9, 102, '2024-01-05');
INSERT INTO purchases VALUES (10, 103, '2024-01-06');
INSERT INTO purchases VALUES (11, 102, '2024-01-06');
INSERT INTO purchases VALUES (12, 107, '2024-01-07');
```

Solution -



```
   customer_id
1  101
2  102
```

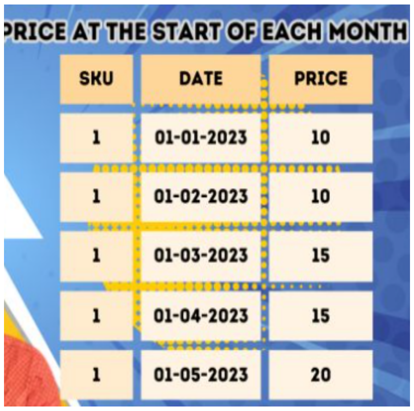
4/03/2024

1. **You are given the price of each sku whenever there is a change in price. Write an SQL to find the price at the start of each month.**

Table script :

```
create table sku
(
sku_id int,
price_date date ,
price int
);
delete from sku;
insert into sku values
(1,'2023-01-01',10)
,(1,'2023-02-15',15)
,(1,'2023-03-03',18)
,(1,'2023-03-27',15)
,(1,'2023-04-06',20);
```

SOLUTION -

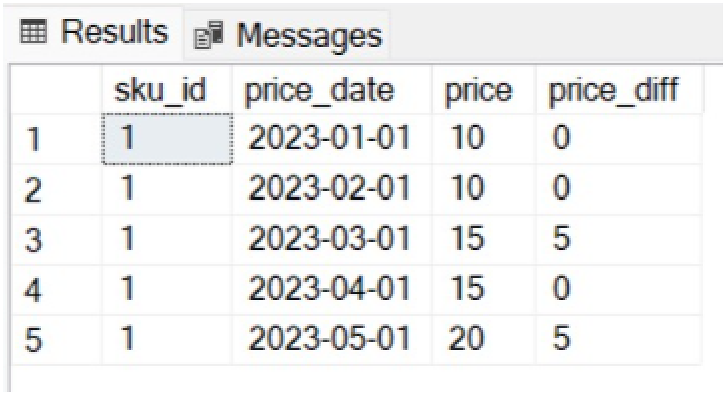


PRICE AT THE START OF EACH MONTH

SKU	DATE	PRICE
1	01-01-2023	10
1	01-02-2023	10
1	01-03-2023	15
1	01-04-2023	15
1	01-05-2023	20

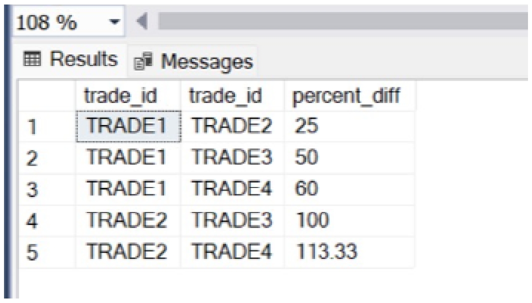
5.Write an SQL query **to find the price at the start of each month & calculate the difference from the previous month's start date**.

Table Script:
create table sku (sku_id int,
price_date date ,
price int
);
insert into sku values (1,'2023-01-01',10)
,(1,'2023-02-15',15)
,(1,'2023-03-03',18)
,(1,'2023-03-27',15)
,(1,'2023-04-06',20);



Results Messages

	sku_id	price_date	price	price_diff
1	1	2023-01-01	10	0
2	1	2023-02-01	10	0
3	1	2023-03-01	15	5
4	1	2023-04-01	15	0
5	1	2023-05-01	20	5



	trade_id	trade_id	percent_diff
1	TRADE1	TRADE2	25
2	TRADE1	TRADE3	50
3	TRADE1	TRADE4	60
4	TRADE2	TRADE3	100
5	TRADE2	TRADE4	113.33

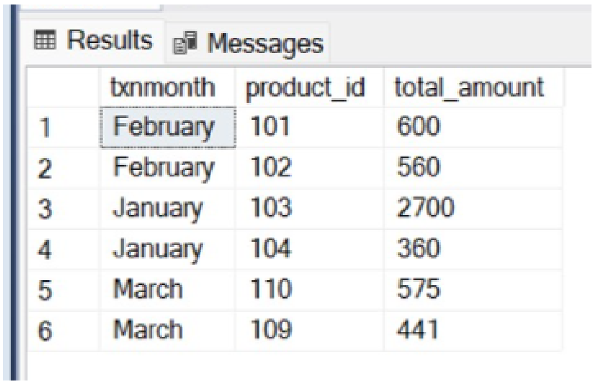
6. Write an SQL query **to find the top 2 selling products for each month in the year 2023**.

Table Script:
CREATE TABLE sales (
sale_id INT PRIMARY KEY,
product_id INT,
sale_date DATE,
quantity INT,
price_per_unit DECIMAL
);

INSERT INTO sales VALUES (1, 101, '2023-01-15', 20, 10.00);
INSERT INTO sales VALUES (2, 101, '2023-01-17', 15, 20.00);
INSERT INTO sales VALUES (3, 103, '2023-01-20', 18, 150.00);
INSERT INTO sales VALUES (4, 101, '2023-01-10', 25, 10.00);
INSERT INTO sales VALUES (5, 104, '2023-01-12', 30, 12.00);

INSERT INTO sales VALUES (6, 102, '2023-02-15', 28, 20.00);
INSERT INTO sales VALUES (7, 102, '2023-02-17', 22, 17.00);
INSERT INTO sales VALUES (8, 101, '2023-02-05', 19, 10.00);
INSERT INTO sales VALUES (9, 103, '2023-02-07', 24, 22.00);
INSERT INTO sales VALUES (10,101, '2023-02-11', 30, 20.00);

INSERT INTO sales VALUES (11, 107, '2023-03-15', 16, 18.00);
INSERT INTO sales VALUES (12, 108, '2023-03-18', 14, 19.00);
INSERT INTO sales VALUES (13, 109, '2023-03-20', 21, 21.00);
INSERT INTO sales VALUES (14, 110, '2023-03-22', 25, 23.00);
INSERT INTO sales VALUES (15, 111, '2023-03-25', 20, 20.00);



Results | Messages

	txnmonth	product_id	total_amount
1	February	101	600
2	February	102	560
3	January	103	2700
4	January	104	360
5	March	110	575
6	March	109	441

5/03/2024

1. **Find how many products falls into customer budget along with list of products.**

– - In case of clash choose the less costly product .

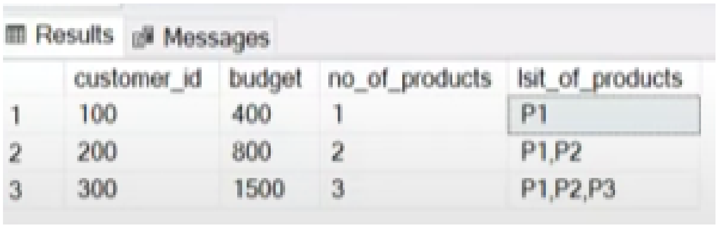
Table Script:

```
create table products
(
product_id varchar(20) ,
cost int
);
insert into products values ('P1',200),('P2',300),('P3',500),('P4',800);

create table customer_budget
(
customer_id int,
budget int
);

insert into customer_budget values (100,400),(200,800),(300,1500);
```

Solution -



Results | Messages

	customer_id	budget	no_of_products	lsit_of_products
1	100	400	1	P1
2	200	800	2	P1,P2
3	300	1500	3	P1,P2,P3

2. **Find the origin and final destination for each cid.**

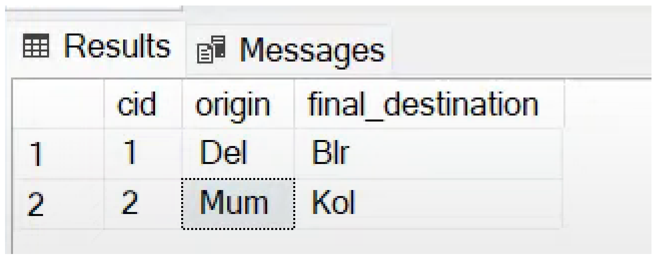
Table script :

```
CREATE TABLE flights
```

```
(
   cid VARCHAR(512),
   fid VARCHAR(512),
   origin VARCHAR(512),
   Destination VARCHAR(512)
);

INSERT INTO flights (cid, fid, origin, Destination) VALUES ('1', 'f1', 'Del', 'Hyd');
INSERT INTO flights (cid, fid, origin, Destination) VALUES ('1', 'f2', 'Hyd', 'Blr');
INSERT INTO flights (cid, fid, origin, Destination) VALUES ('2', 'f3', 'Mum', 'Agra');
INSERT INTO flights (cid, fid, origin, Destination) VALUES ('2', 'f4', 'Agra', 'Kol');
```

Solution -



	cid	origin	final_destination
1	1	Del	Blr
2	2	Mum	Kol

```
SELECT
  distinct a.cid,
   b.origin AS origin,
   c.destination AS destination
FROM
   flights AS a
JOIN
   flights AS b ON a.cid = b.cid AND b.fid = (
      SELECT MIN(fid) FROM flights WHERE cid = a.cid
   )
JOIN
   flights AS c ON a.cid = c.cid AND c.fid = (
      SELECT MAX(fid) FROM flights WHERE cid = a.cid
   );
```

```
select distinct a.cid, b.origin, c.destination from flights as a join
flights as b
on a.cid = b.cid and b.fid =
(select min(fid) from flights where cid = a.cid)
join flights as c
on a.cid = c.cid and c.fid =
(select max(fid) from flights where cid = c.cid)
```

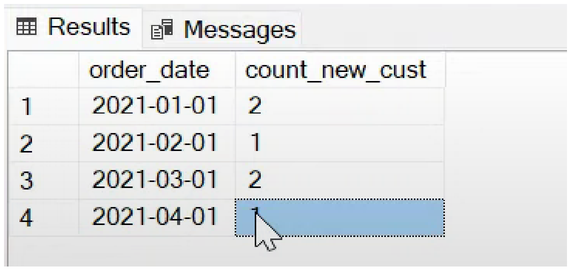
3. **Find the count of new customers added in each month.**

Table script :

```
CREATE TABLE sales
(
   order_date date,
   customer VARCHAR(512),
   qty INT
);

INSERT INTO sales (order_date, customer, qty) VALUES ('2021-01-01', 'C1', '20');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-01-01', 'C2', '30');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-02-01', 'C1', '10');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-02-01', 'C3', '15');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-03-01', 'C5', '19');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-03-01', 'C4', '10');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-04-01', 'C3', '13');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-04-01', 'C5', '15');
INSERT INTO sales (order_date, customer, qty) VALUES ('2021-04-01', 'C6', '10');
```

Solution -



	order_date	count_new_cust
1	2021-01-01	2
2	2021-02-01	1
3	2021-03-01	2
4	2021-04-01	1

4. **Person and type is the column names and it is the input table**

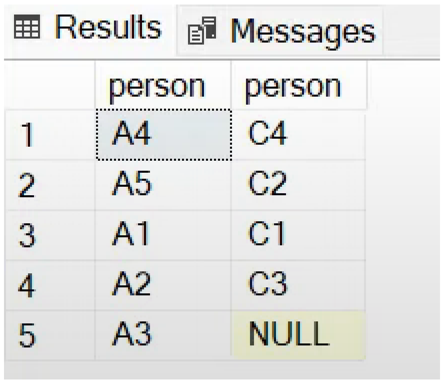
Like these are adult and child in the table and they are going for a fair and they have a ride on some jhoola, so one adult can go with one child and in last one adult will be alone .

Table script :

```
create table family
(
person varchar(5),
type varchar(10),
age int
);
delete from family ;
insert into family values ('A1','Adult',54)
```

,('A2','Adult',53),('A3','Adult',52),('A4','Adult',58),('A5','Adult',54),('C1','Child',20),('C2','Child',19),('C3','Child',22),('C4','Child',15);

SOLUTION -



Results Messages

	person	person
1	A4	C4
2	A5	C2
3	A1	C1
4	A2	C3
5	A3	NULL

6/03/2024

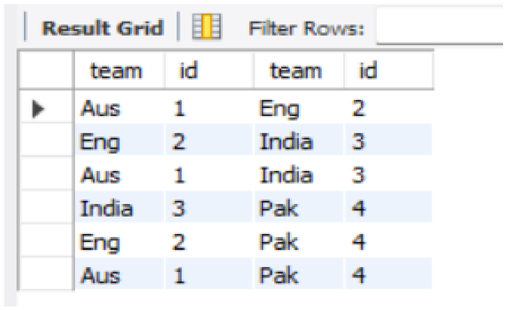
1. **List all the matches between teams, if matches are played once.**

Table script :
CREATE TABLE match_table (team varchar(20));

INSERT INTO match_table (team) VALUES ('India'), ('Pak'), ('Aus'), ('Eng');

SOLUTION -

```
with cte as (
select * , row_number() over(order by team asc) as id
from match_table
)
select * from cte as a
join cte as b
on a.team <> b.team
where a.id < b.id
```



Result Grid Filter Rows:

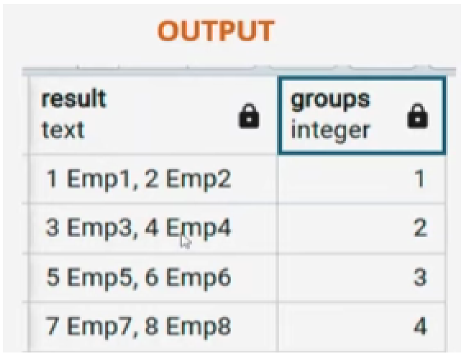
team	id	team	id
Aus	1	Eng	2
Eng	2	India	3
Aus	1	India	3
India	3	Pak	4
Eng	2	Pak	4
Aus	1	Pak	4

2. **write a query to get the output.**

Table script :
CREATE TABLE emp (ID int, NAME varchar(10));

INSERT INTO emp (ID, NAME)
VALUES (1,'Emp1'), (2,'Emp2'), (3,'Emp3'), (4,'Emp4'),
(5,'Emp5'), (6,'Emp6'), (7,'Emp7'), (8,'Emp8');

<u>Solution</u> -


OUTPUT

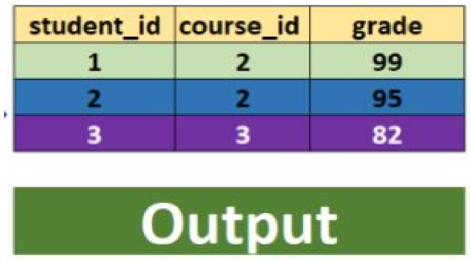
result text	groups integer
1 Emp1, 2 Emp2	1
3 Emp3, 4 Emp4	2
5 Emp5, 6 Emp6	3
7 Emp7, 8 Emp8	4

2. **Write a SQL query to find the highest grade with its corresponding course for each student. In case of a tie, you should find the course with the smallest course_id. The output must be sorted by increasing student_id.**

Table script :
Create table If Not Exists Enrollments
(student_id int,
course_id int,
grade int
);

insert into Enrollments (student_id, course_id, grade) values ('2', '2', '95') ,
('2', '3', '95'),
('1', '1', '90'),
('1', '2', '99'),
('3', '1', '80'),
('3', '2', '75'),
('3', '3', '82');

Solution -



student_id	course_id	grade
1	2	99
2	2	95
3	3	82

Output

```
SELECT
   e1.student_id,
   e1.course_id,
   e1.grade
FROM
   enrollments e1
WHERE
   e1.grade = (SELECT MAX(e2.grade)
            FROM enrollments e2
            WHERE e2.student_id = e1.student_id)
   AND e1.course_id = (SELECT MIN(e3.course_id)
                 FROM enrollments e3
                 WHERE e3.student_id = e1.student_id
                 AND e3.grade = e1.grade)
ORDER BY
   e1.student_id, e1.course_id;
```

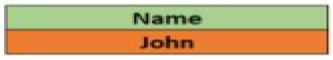
3. **Given the Employee table, write a SQL query that finds out managers with at least 5 direct reportee.**

Table script :

```
Create table If Not Exists Employee (
Id int,
Name varchar(255),
Department varchar(255),
ManagerId int
);

insert into Employee(Id, Name, Department, ManagerId) values
('101', 'John', 'A', Null),
('102', 'Dan', 'A', '101'),
('103', 'James', 'A', '101'),
('104', 'Amy', 'A', '101'),
('105', 'Anne', 'A', '101'),
('106', 'Ron', 'B', '101'),
('107', 'Tony', 'C', '103'),
('108', 'Rocky', 'C', '103');
```

Solution -



Name
John

Output

```
with cte as
(select a.Id, count(a.Id) as cnt from
        Employee as a
        inner join
        Employee as b
        on a.Id = b.ManagerId
        group by a.Id)
select Name from Employee inner join cte
on Employee.Id = cte.Id and cnt >= 5 ;
```

7/03/2024

1. **Write a sql query to find the products whose sales increased every year.**

– include the product_id, product_name, category.

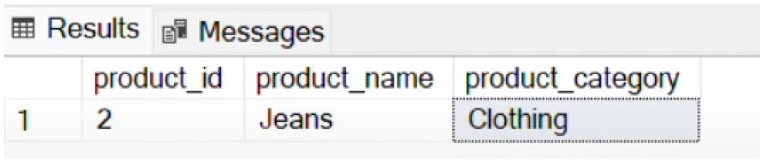
Table script :

```
CREATE TABLE products (
 product_id INT PRIMARY KEY,
 product_name VARCHAR(50),
 category VARCHAR(50)
);

INSERT INTO products (product_id, product_name, category) VALUES
 (1, 'Laptops', 'Electronics'),
 (2, 'Jeans', 'Clothing'),
 (3, 'Chairs', 'Home Appliances');


CREATE TABLE sales (
 product_id INT,
 year INT,
 total_sales_revenue DECIMAL(10, 2),
 PRIMARY KEY (product_id, year),
 FOREIGN KEY (product_id) REFERENCES products(product_id)
);
INSERT INTO sales (product_id, year, total_sales_revenue) VALUES
 (1, 2019, 1000.00),
 (1, 2020, 1200.00),
 (1, 2021, 1100.00),
 (2, 2019, 500.00),
 (2, 2020, 600.00),
 (2, 2021, 900.00),
 (3, 2019, 300.00),
 (3, 2020, 450.00),
 (3, 2021, 400.00);
```

Solution -



Results | Messages

	product_id	product_name	product_category
1	2	Jeans	Clothing

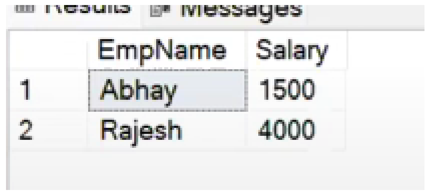
2. **Write an sql to find employees whose salary is greater than their managers salary.**

Table script:

```
CREATE TABLE Employees
(
 EmpID  INT,
 EmpName  VARCHAR(100),
 Salary  INT,
 ManagerID INT
);

INSERT INTO Employees (EmpID,EmpName,Salary,ManagerID)
SELECT 1,'Bala',1000,7
UNION ALL
SELECT 2,'Abhay',1500,1
UNION ALL
SELECT 3,'Lakshman',1500,6
UNION ALL
SELECT 4,'Dileep',2000,7
UNION ALL
SELECT 5,'Rajesh',4000,6
UNION ALL
SELECT 6,'Raviteja',3000,7
UNION ALL
SELECT 7,'Vijay',3500,null
UNION ALL
SELECT 8,'Rakesh',3500,7
```

Solution -


Results Messages

	EmpName	Salary
1	Abhay	1500
2	Rajesh	4000

Solution-1

select * from employees as a, employees as b where a.EmpId = b.ManagerId;
select employees , employee_salary from (
select a.EmpName as manager, b.EmpName as employees, a.Salary as manager_salary, b.Salary as employee_salary
from employees as a,employees as b where a.EmpId = b.ManagerId) as d
where employee_salary > manager_salary ;

16/03/2024

1.Write a query that outputs the name of the credit card, and how many cards were issued in its launch month. The launch month is the earliest record in the monthly_card_issued table for a given card. Order the results starting from the biggest issued amount.

SQL Script:

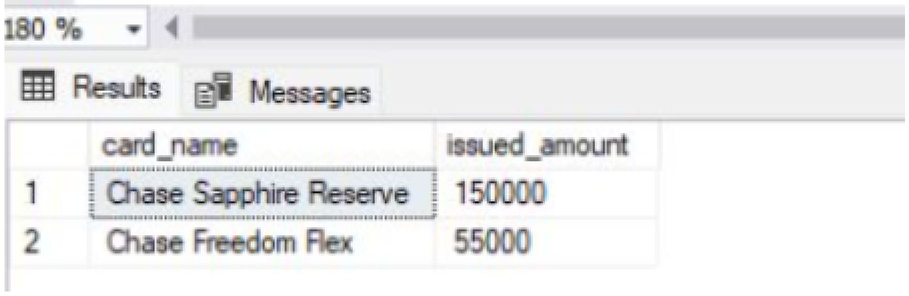
```
CREATE TABLE monthly_card_issued (
  issue_month INTEGER,
  issue_year INTEGER,
  card_name varchar(50),
  issued_amount INTEGER
);

INSERT INTO monthly_card_issued (card_name, issued_amount, issue_month, issue_year)
VALUES
  ('Chase Sapphire Reserve', 160000, 12, 2020),
  ('Chase Sapphire Reserve', 170000, 1, 2021),
  ('Chase Sapphire Reserve', 175000, 2, 2021),
  ('Chase Sapphire Reserve', 180000, 3, 2021),
  ('Chase Freedom Flex', 55000, 1, 2021),
  ('Chase Freedom Flex', 60000, 2, 2021),
  ('Chase Freedom Flex', 65000, 3, 2021),
  ('Chase Freedom Flex', 70000, 4, 2021),
  ('Chase Sapphire Reserve', 150000, 11, 2020);
```

Solution -

```
with cte as(
select card_name, issue_month, issued_amount,
dense_rank() over(partition by card_name order by issue_year, issue_month) rnk
from monthly_card_issued)
select card_name, issued_amount
from cte
where rnk = 1
order by issued_amount desc ;
```

RESULT



```
   card_name               issued_amount
1  Chase Sapphire Reserve  150000
2  Chase Freedom Flex      55000
```

2. Write an SQL query to return the footer values from input table, meaning all the last non null values from each field as shown in expected output.

Table Script:

```
CREATE TABLE footer
(
id INT PRIMARY KEY,
car VARCHAR(20),
length INT,
width INT,
height INT
);
INSERT INTO footer VALUES (1, 'Hyundai Tucson', 15, 6, NULL);
INSERT INTO footer VALUES (2, NULL, NULL, NULL, 20);
INSERT INTO footer VALUES (3, NULL, 12, 8, 15);
INSERT INTO footer VALUES (4, 'Toyota Rav4', NULL, 15, NULL);
INSERT INTO footer VALUES (5, 'Kia Sportage', NULL, NULL, 18);
```

SOLUTION:

```
select * from
(select car from footer where id = (select max(id)from footer where car is not null) ) as a,
(select length from footer where id = (select max(id)from footer where length is not null) )
as b,
(select width from footer where id = (select max(id)from footer where width is not null) )
as c,
(select height from footer where id = (select max(id)from footer where height is not null) )
as d;
```

<u>18/03/2024</u>

1. **Problem Statement**(**Generate salary report**):
Using the given Salary, Income and Deduction tables, first write an sql query to populate the Emp_Transaction table as shown below and then generate a salary report as shown.

Table Script:
create table salary(emp_id int, emp_name varchar(30), base_salary int);
insert into salary values(1, 'Rohan', 5000), (2, 'Alex', 6000), (3, 'Maryam', 7000);

create table income(id int, income varchar(20), percentage int);
insert into income values(1,'Basic', 100), (2,'Allowance', 4), (3,'Others', 6);

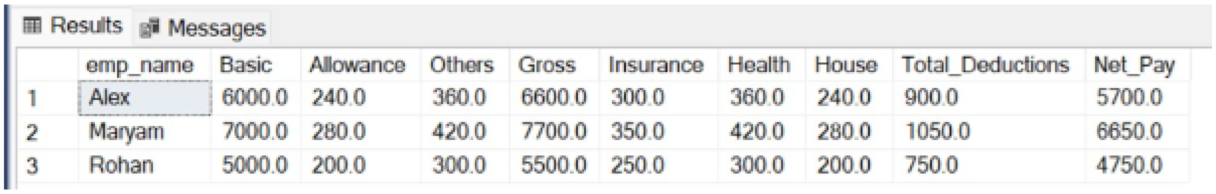
create table deduction(id int, deduction varchar(20), percentage int);
insert into deduction values(1,'Insurance', 5), (2,'Health', 6), (3,'House', 4);

Solution -

```
select a.emp_name, a.Basic,b.Allowance, c.Others,( a.Basic + b.Allowance + c.Others) as
Gross,
d.Insurance, e.Health, f.House, (d.Insurance + e.Health + f.House) as Total_deduction,
((a.Basic + b.Allowance + c.Others)-(d.Insurance + e.Health + f.House)) as Net_pay
from
(select emp_name, round((base_salary * percentage)/100, 1 )as Basic  from salary,
income where id = 1 ) as a
inner join
(select emp_name, round((base_salary * percentage)/100 , 1)as Allowance  from salary,
income where id = 2 ) as b
inner join
(select emp_name, round((base_salary * percentage)/100 , 1)as Others  from salary,
income where id = 3 ) as c
inner join
(select emp_name, round((base_salary * percentage)/100, 1) as Insurance  from salary,
deduction where id = 1 ) as d
inner join
(select emp_name, round((base_salary * percentage)/100 , 1)as Health  from salary,
deduction where id = 2 ) as e
```

```
inner join
(select emp_name, round((base_salary * percentage)/100 , 1)as House  from salary,
deduction where id = 3 ) as f
on a.emp_name = b.emp_name
and a.emp_name= c.emp_name
and a.emp_name= d.emp_name
and a.emp_name= e.emp_name
and a.emp_name= f.emp_name
order by a.emp_name ;
```

OUTPUT



Results | Messages

	emp_name	Basic	Allowance	Others	Gross	Insurance	Health	House	Total_Deductions	Net_Pay
1	Alex	6000.0	240.0	360.0	6600.0	300.0	360.0	240.0	900.0	5700.0
2	Maryam	7000.0	280.0	420.0	7700.0	350.0	420.0	280.0	1050.0	6650.0
3	Rohan	5000.0	200.0	300.0	5500.0	250.0	300.0	200.0	750.0	4750.0

20/03/2024

1.**Problem Statement(Student Performance)**:
You are given **a table having the marks of one student in every test. You have to output the tests in which the student has improved his performance. For a student to improve his performance he has to score more than the previous test. Provide 2 solutions, one including the first test score and second excluding it**.

Table Script:

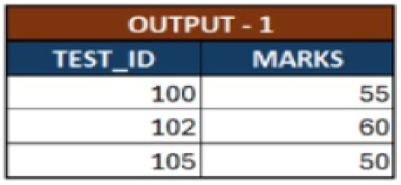
```
create table student_tests(test_id int, marks int);
insert into student_tests values(100, 55);
insert into student_tests values(101, 55);
insert into student_tests values(102, 60);
insert into student_tests values(103, 58);
insert into student_tests values(104, 40);
insert into student_tests values(105, 50);
```

SOLUTION -

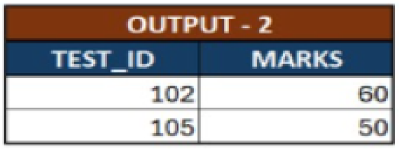
```
-- output1
with cte as (
select *, lag(marks) over() as lag1 from student_tests
)
select test_id, marks  from cte where marks-lag1 > 0 or marks-lag1 is null ;

-- output2
```

```
with cte as (
select *, lag(marks) over() as lag1 from student_tests
)
select test_id, marks from cte where marks-lag1 > 0 ;
```



OUTPUT - 1	
TEST_ID	MARKS
100	55
102	60
105	50



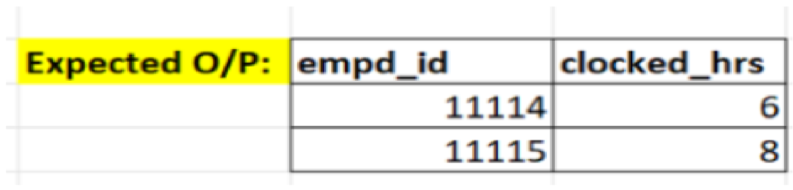
OUTPUT - 2	
TEST_ID	MARKS
102	60
105	50

2. Problem Statement:
Write an SQL query to find total number of clocked hours for each employee(inside the office), flag - I means punch in and O means punch out. Employee can do multiple punch in and punch out in a day. for each punch in there will be a punch out.

You have the input table clocked_hours and you need to derive output table as shown in the picture.

Table Script:
```
create table clocked_hours(
empd_id int,
swipe time,
flag char
);
insert into clocked_hours values
(11114,'08:30','I'),
(11114,'10:30','O'),
(11114,'11:30','I'),
(11114,'15:30','O'),
(11115,'09:30','I'),
(11115,'17:30','O');
```

SOLUTION -
```
with cte1 as (
select empd_id, flag, sum(swipe) as intme
   from clocked_hours group by empd_id, flag having flag = "I"
   ) ,
cte2 as (
select empd_id, flag, sum(swipe)  as outtme
   from clocked_hours group by empd_id, flag having flag = "O"
   )
select cte1.empd_id, hour(time(outtme - intme)) as workedhour
from cte1, cte2 where cte1.empd_id = cte2.empd_id ;
```



Expected O/P:	empd_id	clocked_hrs
	11114	6
	11115	8

3. Problem Statement:

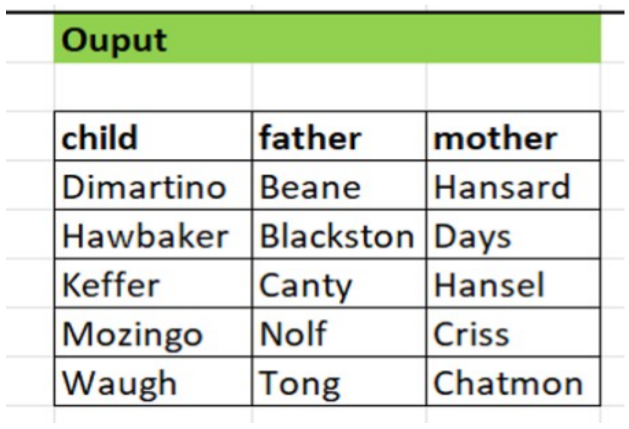
Table Script:

```
create table people(id int primary key not null,
name varchar(20),
gender char(2));

create table relations(c_id int,
p_id int,
FOREIGN KEY (c_id) REFERENCES people(id),
foreign key (p_id) references people(id)
);

insert into people (id, name, gender) values
(107,'Days','F'),
(145,'Hawbaker','M'),
(155,'Hansel','F'),
(202,'Blackston','M'),
(227,'Criss','F'),
(278,'Keffer','M'),
(305,'Canty','M'),
(329,'Mozingo','M'),
(425,'Nolf','M'),
(534,'Waugh','M'),
(586,'Tong','M'),
(618,'Dimartino','M'),
(747,'Beane','M'),
(878,'Chatmon','F'),
(904,'Hansard','F');

insert into relations(c_id, p_id) values
(145, 202),
(145, 107),
(278,305),
(278,155),
(329, 425),
(329,227),
(534,586),
(534,878),
(618,747),
(618,904);
```



Ouput

child	father	mother
Dimartino	Beane	Hansard
Hawbaker	Blackston	Days
Keffer	Canty	Hansel
Mozingo	Nolf	Criss
Waugh	Tong	Chatmon