PRICE TRACKER

Database Management Systems Project

Winter Semester, 2020

GROUP NO. 12

AU1841131 MANSI DOBARIYA
AU1841051 BHUMITI GOHEL
AU1841022 MANAV VAGRECHA

Under the Guidance of
Prof. Shefali Naik

INTRODUCTION:

Price tracking is a cost-effective way to track, compare, and analyze the prices of products in this competitive world. Today, purchasing goods and services online is a prevalent practice among millions of people worldwide. Purchasing online is convenient, easy, and economically advantageous, as multiple e-commerce platforms offer competitive prices. Thus, both sellers and buyers are bound to execute price monitoring, comparison, and analysis. The number of online purchases is constantly growing based on the decreasing price rate through variation given by price tracker App/web.

PROJECT DEFINITION :

This website tracks prices and quantity of products. Each product is included in one Category. FOR EG. product = 'iphone' and Category = 'Electronics And Gadgets' but each such Category contains many products in itself. FOR EG Category = 'Electronics And Gadgets' has many Products = { 'iPhone', 'Laptop', 'Headphones', etc. }. Products can be dependent on Age Groups such as Male, Female and Kids. FOR EG. Few products are only for females where some for Males and some for kids where some products are neutral w.r.t. (independent of) Age groups (FOR EG. Furniture). But for Each age Group, there will be many Products. Each Product can have many Brand FOR EG. product = 'Laptop', Brand = {'Asus', 'Lenovo', 'Dell', etc.}. Each Brand can have many products FOR EG. Brand = 'Sony', Product = {'PlayStation','Smart Phones','Televisions',etc.}. Further, Each Brand can have products having different Categories. FOR EG. Brand = 'Nike' Category = { 'Shoes', 'Clothings', 'Sport items', etc. }. Each Category can contain many products having different Brands. FOR EG. Category = 'Clothings' Brand = { 'Levis', 'Under Armour', 'Armaani', etc. }. Every website contains a specific stock of product with it where they can also note the number of Sold Products i.e. Quantities Sold. Each product can have many Quantities. Every Product has Prices which changes whenever there is a Sale or Festive but the M.R.P. almost always remains constant. Few Products

may have Offers with them where the users can be benefitted and extra discounts can be earned.

A user uses this website and it may register to which his / hers' history can be saved. Thus, users can have their search history and other activity tracked. Each (registered)user of the website can have a lot of Activity and Search History data. Each history can correspond to more than one product searched for or Editted by the Admin.

ENTITY-RELATIONSHIP DIAGRAM:

ENTITIES

Primary Key : product(product_id), brand(brand_id), user_detail(user_id), category(category_id), group_detail(age_group_id) ,product_details (product_id +brand_id + category_id +age_group_id),history(history_id) ;

Cardinality:

product - category(M-1)
product - brand (M-N)
brand - category (M-N)
product - group_detail (M-N)
product - offer(1-M)
product - quantity (1-M)
product - price_details (1-M)
history - user_detail (1-M)

E-R DIAGRAM


availlibility
MRP
product_image
product_id + brand_id + category_id + group_id
age_group_id
may have
group_detail
M
product_detail
M
contains
ratings
current_price
full_product_name
age_group
brand
contains
brand_name
category_name
1
brand_id
category
contains
category_id
has
product_name
offer_detail
1M
offer
product_id
may have
1
product
quantity_new
has
product_id
M
quantity
description
search_word
quantity_old
must have
history_id
search_history
email_id
M
price_details
time_src
password
N
1
has
user_detail
user_id
discount
description
1
contains
date_of_price_changing
price
old_values
current_time
audit_track_table
new_value
1
query_
table_name

DATA DICTIONARY:

Audit_table_track

Column	Type	Null	Default	Links to	Comments	MIME
user_id	varchar(50)	No		user_details -> user_id		
table_name	varchar(50)	No				
old_value	varchar(50)	No				
new_value	varchar(50)	No				
query	varchar(50)	No				
crt_time	datetime	No	CURRENT_TIMESTAMP			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
user_id	BTREE	Yes	No	user_id	0	A	No	

Brand

Column	Type	Null	Default	Links to	Comments	MIME
brand_id *(Primary)*	int(11)	No				
brand_name	varchar(1000)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	brand_id	52	A	No	

Category

Column	Type	Null	Default	Links to	Comments	MIME
category_id *(Primary)*	int(11)	No				
category_name	varchar(1000)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	category_id	7	A	No	

Deals

Column	Type	Null	Default	Links to	Comments	MIME
product_id	int(11)	No		product -> product_id		
deals_details	varchar(500)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
product_id	BTREE	No	No	product_id	6	A	No	

Group_detail

Column	Type	Null	Default	Links to	Comments	MIME
age_group_id *(Primary)*	int(10)	No				
age_detail	varchar(15)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment

PRIMARY	BTREE	Yes	No	age_group_id	4	A	No	
age_detail	BTREE	Yes	No	age_detail	4	A	No	

Price_details

Column	Type	Null	Default	Links to	Comments	MIME
product_id	int(11)	No		product -> product_id		
date_of_price_changing	date	No				
price	float	No				
description	varchar(50)	No				
discount	float	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
product_id	BTREE	Yes	No	product_id	162	A	No	
				date_of_price_changing	810	A	No	

Product

Column	Type	Null	Default	Links to	Comments	MIME
product_id *(Primary)*	int(11)	No				
product_name	varchar(1000)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	product_id	81	A	No	

Product_details

Column	Type	Null	Default	Links to	Comments	MIME
product_id	int(11)	No	0	product -> product_id		
category_id	int(11)	No		category -> category_id		
age_group_id	int(10)	Yes	*NULL*	group_detail -> age_group_id		
brand_id	int(11)	No		brand -> brand_id		
full_product_name	varchar(500)	No				
available	char(1)	No				
mrp	float	No				
images	blob	No				
rating	int(5)	No				
current_price	float	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
prod_cat_brd_age_id_uniq	BTREE	Yes	No	brand_id	71	A	No	
				product_id	71	A	No	
				category_id	71	A	No	
				age_group_id	71	A	Yes	
product_id	BTREE	No	No	product_id	71	A	No	
category_id	BTREE	No	No	category_id	14	A	No	

age_group_id	BTREE	No	No	age_group_id	10	A	Yes	
brand_id	BTREE	No	No	brand_id	71	A	No	

Quantity

Column	Type	Null	Default	Links to	Comments	MIME
product_id	int(11)	No		product -> product_id		
quantity_new	int(11)	No				
quantity_old	int(11)	No				
demand	float	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
product_id	BTREE	Yes	No	product_id	81	A	No	

Search_history

Column	Type	Null	Default	Links to	Comments	MIME
history_id *(Primary)*	int(11)	No				
user_id	varchar(50)	No		user_details -> user_id		
search_word	varchar(2000)	No				

description	varchar(2000)	No				
time_src	datetime	No	CURRENT_TIMESTAMP			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	history_id	0	A	No	
user_id	BTREE	No	No	user_id	0	A	No	

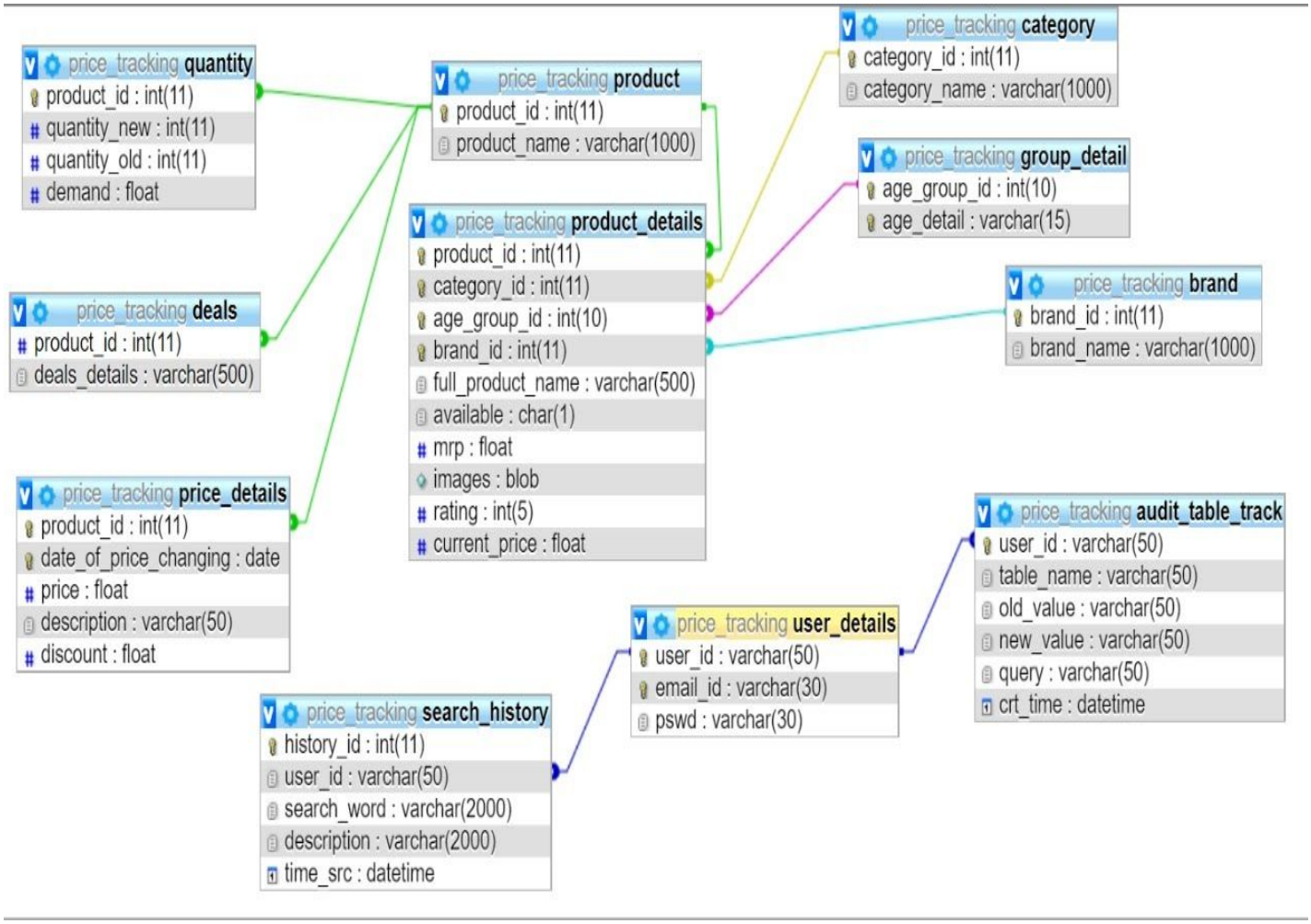
User_details

Column	Type	Null	Default	Links to	Comments	MIME
user_id *(Primary)*	varchar(50)	No				
email_id	varchar(30)	No				
pswd	varchar(30)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	user_id	0	A	No	
email_id	BTREE	Yes	No	email_id	0	A	No	

CLASS DIAGRAM:



STORED PROCEDURES:

Search Product:

```
DELIMITER$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `search_prod`(IN `xyz` VARCHAR(50))
   NO SQL
BEGIN
SELECT search_product() as sp;
SELECT DISTINCT product_details.product_id ,product_details.full_product_name
,product_details.mrp, product_details.current_price FROM product_details WHERE
LCASE(product_details.full_product_name)  like LCASE(concat('%',xyz,'%'));
END$$
DELIMITER ;
```

Show brand:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `show_brand`(IN `cat_id` INT)
   NO SQL
bgn1 : BEGIN
DECLARE b_id int default 1000;
DECLARE c_id int default 100;
DECLARE x1 int default 0;
DECLARE c_brand_cat CURSOR FOR SELECT Distinct brand_id, category_id from
product_details;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET x1 = 1;
OPEN c_brand_cat;
set x1 = 0;
        FETCH c_brand_cat into b_id, c_id;
        while(x1=0) DO
                if (c_id = cat_id) then
                        bgn2 : BEGIN
                        DECLARE x2 int default 0;
                        DECLARE id_b int default 1000;
```

```
                DECLARE name varchar(30);
                DECLARE c_brand CURSOR for SELECT * from brand;
                DECLARE CONTINUE HANDLER FOR NOT FOUND SET x2 = 1;
                OPEN c_brand;
            set x2 = 0;
                FETCH c_brand into id_b, name;
                    x2while : while(x2 = 0) do
                        if( id_b = b_id) THEN
                            select b_id, name, c_id;
                            LEAVE x2while;
                        end if;
                    FETCH c_brand into id_b, name;
                     end while x2while;
                CLOSE c_brand;
                END bgn2;
             end if;
            FETCH c_brand_cat into b_id, c_id;
        end while;
CLOSE c_brand_cat;
END bgn1$$
DELIMITER ;
```

DEMAND:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `demand`()
bgn1: BEGIN
DECLARE prod_id int DEFAULT 0;
DECLARE x1 int DEFAULT 0;
DECLARE c1 CURSOR for SELECT distinct product_id from price_details;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET x1 = 1;
open c1;
 set x1 = 0;
        FETCH c1 into prod_id;
        w1 : while(x1=0) DO
        bgn2 : BEGIN
                DECLARE prid int;
                DECLARE dateX_new date;
                DECLARE pr_new float;
                DECLARE dateX_old date;
```

```
		DECLARE pr_old float;
		DECLARE quan_old int;
		DECLARE quan_new int;
		DECLARE elas_demand float;
		DECLARE c2 CURSOR for select quantity.product_id, date_of_price_changing, price,
quantity_old, quantity_new from price_details, quantity where quantity.product_id = prod_id
AND price_details.product_id = prod_id ORDER by date_of_price_changing DESC LIMIT 2;
		open c2;
			FETCH c2 into prid, dateX_new, pr_new, quan_old, quan_new;
			FETCH c2 into prid, dateX_old, pr_old, quan_old, quan_new;
			set elas_demand = (quan_new - quan_old)*100/(pr_new - pr_old);
			Select prid, elas_demand;
			Update quantity set demand = elas_demand where quantity.product_id =
prid;
		close c2;
		end bgn2;
	FETCH c1 into prod_id;
	end while w1;
END bgn1$$
DELIMITER ;
```

Sell :

```
DELIMITER$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sell`()
NO SQL
bgn1 : BEGIN
DECLARE done INT DEFAULT 0;
DECLARE dn INT DEFAULT 0;
DECLARE Total FLOAT DEFAULT 0;
DECLARE id int DEFAULT 0;
DECLARE di int DEFAULT 0;
DECLARE name VARCHAR(30);
DECLARE c1 CURSOR FOR SELECT
   ((quantity_old - quantity_new)*100)/quantity_old , product_id from quantity;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
OPEN c1;
SET done = 0;
	FETCH c1 INTO Total, id;
	reloop : while done = 0 do
```

```
        bgn2 : BEGIN
       DECLARE c2 CURSOR FOR SELECT product_id, full_product_name from product_details;
       DECLARE CONTINUE HANDLER FOR NOT FOUND SET dn = 1;
       OPEN c2;
       SET dn = 0;
        FETCH c2 into di,name;
        repool: while dn=0 DO
                        IF (di = id) then
                                SELECT di, name, total;
                                LEAVE repool;
                        END IF;
                        FETCH c2 into di, name;
        END WHILE repool;
        CLOSE c2;
        END bgn2;
        FETCH c1 into total, id;
      END while reloop;
CLOSE c1;
END bgn1$$
DELIMITER;
```

Find current price:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `fetch_cprice`()
bgn1 : BEGIN

DECLARE x1 int default 0;
DECLARE pid int DEFAULT 0;
DECLARE dateX date;
DECLARE c1 CURSOR FOR select max(date_of_price_changing), product_id from price_details
group by product_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET x1 = 1;

open c1;
set x1 = 0;
        FETCH c1 into dateX, pid;
        yw : while(x1=0) DO
        bgn2 : Begin
        Declare x2 int;
```

```
        Declare id int;
        Declare prx float;
        Declare xdate date;
        Declare c2 Cursor for select product_id,price,date_of_changing_price from price_details;
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET x2 = 1;
        open c2;
        set x2 = 0;
                FETCH c2 into id, prx, xdate;
                xw : while (x2=0) DO
                if(id = pid AND xdate = dateX) then
                        UPDATE product_details set current_details = prx where product_id = pid;
                       leave xw;
                end if;
                FETCH c2 into id, prx, xdate;
                 end while xw;
        END bgn2;
        FETCH c1 into dateX, pid;
        end while yw;
END bgn1$$
DELIMITER ;
```

SORTING :

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `sort_method`(IN `num` INT, IN `ind` INT)
   NO SQL
BEGIN

if( num = 1 and ind =1) THEN
select DISTINCT product_details.product_id,product_details.full_product_name,
product_details.mrp ,product_details.current_price, product_details.rating from product_details,
price_details where price_details.product_id = product_details.product_id order by
product_details.current_price DESC;

ELSEIF ( num = 1 and ind = 2)THEN
select DISTINCT product_details.product_id,product_details.full_product_name,
product_details.mrp ,product_details.current_price, product_details.rating from product_details,
price_details where price_details.product_id = product_details.product_id order by
product_details.current_price asc;

ELSEIF ( num = 2 and ind = 1)THEN
```

```
select DISTINCT product_details.product_id,product_details.full_product_name,
product_details.mrp ,product_details.current_price, product_details.rating from product_details,
price_details where price_details.product_id = product_details.product_id order by
product_details.full_product_name DESC;

ELSEIF ( num = 2 and ind = 2) THEN

select DISTINCT product_details.product_id,product_details.full_product_name,
product_details.mrp ,product_details.current_price, product_details.rating from product_details,
price_details where price_details.product_id = product_details.product_id order by
product_details.full_product_name asc;

ELSEIF (num = 3 and ind = 1) THEN
select product_details.product_id, product_details.full_product_name, product_details.mrp
,product_details.current_price , product_details.rating from product_details inner join
price_details on price_details.product_id = product_details.product_id where
date_of_price_changing in(select max(date_of_price_changing) from price_details group by
product_id) ORDER by price_details.discount DESC;

ELSEIF (num = 3 and ind =2) THEN
select product_details.product_id, product_details.full_product_name, product_details.mrp
,product_details.current_price , product_details.rating from product_details inner join
price_details on price_details.product_id = product_details.product_id where
date_of_price_changing in(select max(date_of_price_changing) from price_details group by
product_id) ORDER by price_details.discount asc;

ELSEIF( num = 4 and  ind = 1) THEN
select DISTINCT product_details.product_id,product_details.full_product_name,
product_details.mrp ,product_details.current_price, product_details.rating from product_details,
price_details order by product_details.rating DESC ;

ELSEIF (num = 4 and ind = 2) THEN
select DISTINCT product_details.product_id,product_details.full_product_name,
product_details.mrp ,product_details.current_price ,product_details. rating from product_details,
price_details order by product_details.rating ASC;

ELSEIF ( num = 5 and ind = 1) THEN
select DISTINCT product_details.product_id, product_details.full_product_name,
product_details.mrp ,product_details.current_price ,product_details.rating from product_details,
quantity where product_details.product_id = quantity.product_id order by quantity.demand desc;

ELSEIF ( num = 5 and ind =2) THEN
```

```
select DISTINCT product_details.product_id, product_details.full_product_name,
product_details.mrp ,product_details.current_price , product_details.rating from product_details ,
quantity where product_details.product_id = quantity.product_id order by quantity.demand asc;

end if;
End $$

DELIMITER ;
```

FILTERING :

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `filter_method`(IN `num` INT, IN `name`
VARCHAR(20), IN `min` INT, IN `max` INT)
   NO SQL
   DETERMINISTIC
BEGIN

if( num = 1 ) then
select DISTINCT product_details.product_id
,product_details.full_product_name,product_details.mrp,product_details.current_price
,product_details.rating from product_details , brand  where product_details.brand_id =
brand.brand_id and brand.brand_name = name;

ELSEIF (num = 2) then
 select DISTINCT product_details.product_id
,product_details.full_product_name,product_details.mrp,product_details.current_price
,product_details.rating from product_details , product where product_details.current_price
between min and max;

ELSEIF (num = 3) then
select DISTINCT product_details.product_id
,product_details.full_product_name,product_details.mrp,product_details.current_price
,product_details.rating from product_details, deals where product_details.product_id =
deals.product_id;


ELSEIF (num =4) THEN
select DISTINCT product_details.product_id
,product_details.full_product_name,product_details.mrp,product_details.current_price
```

```
,product_details.rating from product_details,price_details where  product_details.product_id =
price_details.product_id and price_details.description = name;


end if;
end$$
DELIMITER ;
```

Track Tables :

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `track_table`(IN `t_name` VARCHAR(50), IN
`type_of_query` VARCHAR(10))
   NO SQL
BEGIN
if(t_name ="product")then
          IF (type_of_query= "insert") THEN
    insert INTO audit_table_track (TABLE_NAME,new_value,query) VALUES
(t_name,new.product_id,type_of_query);
          ELSEIF (type_of_query="update") THEN
    insert into audit_table_track (TABLE_NAME,old_value,new_value,query) VALUES
(t_name,old.product_id,new.product_name,type_of_query);
          ELSE
   insert into audit_table_track (TABLE_NAME,old_value,query) VALUES
(t_name,old.product_id,type_of_query);
          end if;

ELSEIF(t_name ="brand")then
          IF (type_of_query= "insert") THEN
    insert INTO audit_table_track (TABLE_NAME,new_value,query) VALUES
(t_name,new.brand_id,type_of_query);
          ELSEIF (type_of_query="update") THEN
    insert into audit_table_track (TABLE_NAME,old_value,new_value,query) VALUES
(t_name,old.brand_id,new.brand_name,type_of_query);
          ELSE
   insert into audit_table_track (TABLE_NAME,old_value,query) VALUES
(t_name,old.brand_id,type_of_query);
          end if;

ELSEIF(t_name ="category") THEN
          IF (type_of_query = "insert") THEN
```

```
    insert INTO audit_table_track (TABLE_NAME,new_value,query) VALUES
(t_name,new.category_id,type_of_query);
          ELSEIF (type_of_query="update") THEN
    insert into audit_table_track (TABLE_NAME,old_value,new_value,query) VALUES
(t_name,old.category_id,new.category_name,type_of_query);
          ELSE
   insert into audit_table_track (TABLE_NAME,old_value,query) VALUES
(t_name,old.category_id,type_of_query);
          end if;

ELSEIF(t_name="group_detail")THEN
          IF (type_of_query = "insert") THEN
    insert INTO audit_table_track (TABLE_NAME,new_value,query) VALUES
(t_name,new.age_group_id,type_of_query);
          ELSEIF (type_of_query="update") THEN
    insert into audit_table_track (TABLE_NAME,old_value,new_value,query) VALUES
(t_name,old.age_group_id,new.age_group,type_of_query);
          ELSE
   insert into audit_table_track (TABLE_NAME,old_value,query) VALUES
(t_name,old.age_group_id,type_of_query);
          end if;

END if;
END$$
DELIMITER ;
```

STORED FUNCTIONS:

Product Search:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `search_product`() RETURNS int(11)
    NO SQL
BEGIN
DECLARE pid INT DEFAULT 0;
DECLARE pn varchar(100);
DECLARE cn varchar(50);
DECLARE bn varchar(50);
DECLARE con varchar(500);
DECLARE  b INT DEFAULT 0;
DECLARE  c1 CURSOR for SELECT DISTINCT product_details.product_id,product.product_name,
category.category_name,brand.brand_name FROM product ,category ,brand ,product_details
where product_details.product_id = product.product_id and product_details.category_id =
category.category_id and product_details.brand_id = brand.brand_id;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET b = 1;
 OPEN c1;
 SET b = 0;
            FETCH c1 INTO pid ,pn,cn,bn;
            WHILE b = 0 DO
            set con =concat(bn,' ',pn,' ',cn);
             update product_details set full_product_name = con  where
product_details.product_id = pid;
             FETCH c1 INTO pid ,pn,cn,bn;
```

```
        END WHILE;
CLOSE c1;
RETURN 0;
end$$
DELIMITER ;
```

Sign up:

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION`check_user_id`(`userid` INT(30)) RETURNS
char(1) CHARSET utf8
   NO SQL
 DETERMINISTIC
BEGIN
DECLARE uid varchar(30);
DECLARE x INT DEFAULT 0;
DECLARE i INT DEFAULT 0;
DECLARE n INT DEFAULT 0;
DECLARE temp char(1);
DECLARE c_check CURSOR FOR SELECT user_detail.user_id FROM user_detail;
SELECT COUNT(user_detail.user_id) FROM user_detail INTO n;
OPEN c_check;
SET i = 0;
WHILE i<n DO
        FETCH c_check into uid;
        IF uid = userid THEN
                set x = x+1;
        END IF;
        SET i = i + 1;
END WHILE;
CLOSE c_check;

IF x = 1 THEN
        set temp = 'Y';
ELSE
        set temp = 'N';
END IF;
```

```
return temp;
END$$
DELIMITER ;
```

Check log in:

```
DELIMITER$$
CREATE DEFINER=`root`@`localhost` FUNCTION `check_login`(`userid` VARCHAR(30), `pass`
VARCHAR(20)) RETURNS char(1) CHARSET latin1
BEGIN
DECLARE i INT DEFAULT 0;
DECLARE n INT DEFAULT 0;
DECLARE uid varchar(30);
DECLARE psd varchar(20);
DECLARE x INT DEFAULT 0;
DECLARE c_check CURSOR FOR SELECT user_detail.user_id, user_detail.pswd FROM user_detail;
SELECT COUNT(user_detail.user_id) FROM user_detail INTO n;
OPEN c_check;
SET i = 0;
WHILE i<n DO
        FETCH c_check into uid, psd;
        IF  uid = userid and pass = psd THEN
                set x = x+1;
        end if;
        SET i = i + 1;
END WHILE;
CLOSE c_check;

IF x = 1 then
        return 'Y';
ELSE
        return 'N';
END IF;
END$$
DELIMITER ;
```

TRIGGERS:

On Sign Up:

```
DELIMITER$
CREATE TRIGGER trig_register BEFORE insert on user_detail
FOR EACH ROW
BEGIN
	IF check_user_id(new.user_id)='Y' THEN
		msg = concat('Error: User Exists...Try new User id');
		signal sqlstate '45002' set message_text = msg;
	ELSE
		INSERT into logger(content) values ("The user details are successfully registered..");
	END IF;
END$
DELIMITER ;
```

Price Check:

```
DELIMITER$$
CREATE TRIGGER `check_price` BEFORE INSERT ON `dummy2`
FOR EACH ROW
BEGIN
DECLARE msg varchar(128);
DECLARE n int  DEFAULT 0;
SELECT mrp from product_details WHERE PRODUCT_ID =NEW.PRODUCT_ID into n;
	 IF new.price >  n then
		set msg = concat('Error: Price is more than MRP...');
		signal sqlstate '45001' set message_text = msg;
	ELSE
	--Otherwise calculate discount on this product
	    INSERT INTO price_details(product_id, date_of_price_changing, price, description,
discount) VALUES (new.product_id, new.date_of_price_changing, new.price, new.description,
(n -new.price)*100/n) ;
	END IF;
```

```
END$$
DELIMITER ;
```

Track tables through triggers

```
CREATE TRIGGER `track_brand_delete` AFTER DELETE ON `brand`
 FOR EACH ROW BEGIN
        call track_table("brand","delete");
END

CREATE TRIGGER `track_brand_insert` AFTER INSERT ON `brand`
 FOR EACH ROW BEGIN
        call track_table("brand","insert");
END

CREATE TRIGGER `track_brand_update` AFTER UPDATE ON `brand`
 FOR EACH ROW BEGIN
        call track_table("brand","update");
END

CREATE TRIGGER `track_category_delete` AFTER DELETE ON `category`
 FOR EACH ROW BEGIN
        call track_table("category","delete");
END

CREATE TRIGGER `track_category_insert` AFTER INSERT ON `category`
 FOR EACH ROW BEGIN
        call track_table("category","insert");
END

CREATE TRIGGER `track_category_update` AFTER UPDATE ON `category`
 FOR EACH ROW BEGIN
        call track_table("category","update");
END

CREATE TRIGGER `track_group_delete` AFTER DELETE ON `group_detail`
 FOR EACH ROW BEGIN
        call track_table("group_detail","delete");
end

CREATE TRIGGER `track_group_insert` AFTER INSERT ON `group_detail`
 FOR EACH ROW BEGIN
```

```
        call track_table("group_detail","insert");
end

CREATE TRIGGER `track_group_update` AFTER UPDATE ON `group_detail`
 FOR EACH ROW BEGIN
        call track_table("group_detail","update");
end

CREATE TRIGGER `track_product_delete` AFTER DELETE ON `product`
 FOR EACH ROW BEGIN
        call track_table("product","delete");
END

CREATE TRIGGER `track_product_insert` AFTER INSERT ON `product`
 FOR EACH ROW BEGIN
call track_table("product","insert");
END

CREATE TRIGGER `track_product_update` AFTER UPDATE ON `product`
 FOR EACH ROW BEGIN
        call track_table("product","update");
END
```

FEW Example of calling Functions and Procedure Using PHP:

```php
if(isset($_POST['sort']) AND isset($_POST['gender'])){
                        if($_POST['sort'] == "Sort by Prices"){
                            if($_POST['gender'] == 0)
                                $sql = "CALL sorting_method(1, 2);";
```

```
                        else
                            $sql = "CALL sorting_method(1, 1);";
                    }else if($_POST['sort'] == "Sort by Rating"){
                        if($_POST['gender'] == 0)
                            $sql = "CALL sorting_method(4, 2);";
                        else
                            $sql = "CALL sorting_method(4, 1);";
                    }else if($_POST['sort'] == "Sort by Discount"){
                        if($_POST['sort']==0)
                            $sql = "CALL sorting_method(3, 2);";
                        else
                            $sql = "CALL sorting_method(3, 1);";
                    }else if($_POST['sort'] == "Sort by Sell"){
                        if($_POST['sort']==0)
                            $sql = "CALL sorting_method(5, 2);";
                        else
                            $sql = "CALL sorting_method(5, 1);";
                    }else if($_POST['sort'] == "Sort by Name"){
                        if($_POST['gender']==0)
                            $sql = "CALL sorting_method(2, 2);";
                        else
                            $sql = "CALL sorting_method(2, 1);";
                    }else{
                        $sql = "SELECT product_id, full_product_name, mrp,
current_price from product_details";
                    }
                }else{
                      $sql = "SELECT product_id, full_product_name, mrp,
current_price from product_details";
                  }


--------------------------------------------------------------------------------



                if (isset($_POST['searchX'])){
```

```
                $prod_name = $_POST['searchX'];
            }else{
                $prod_name = NULL;
            }

            $sql="CALL search_prod('".$prod_name."')";
```

```
                if(isset($_POST['cat_names'])){
                    $cat_num = $_POST['cat_names'];
                }else{
                    $cat_num = NULL;
                }
                echo $cat_num;
                $d = array();
                $d = explode("cat_", $cat_num);

                $select_brand = "CALL SHOW_BRAND(".(int)$d[0].");";
```

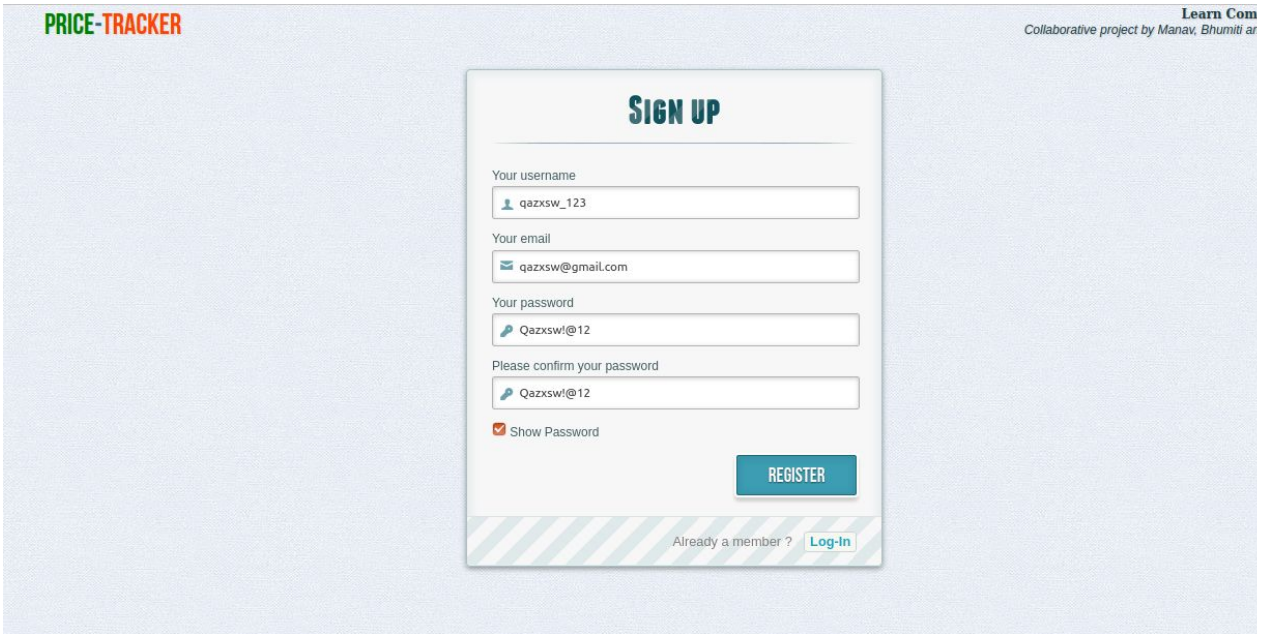
```
if(isset($_POST['submit-login'])){
       $check_in = $_COOKIE['check_lin'];
       $userid_in = $_POST['username'];
       $pass_in = $_POST['password'];

     $sql = "SELECT check_login('".$userid_in."', '".$pass_in."') AS
check_login;";
 }
```

Languages Used And basic Requirements:

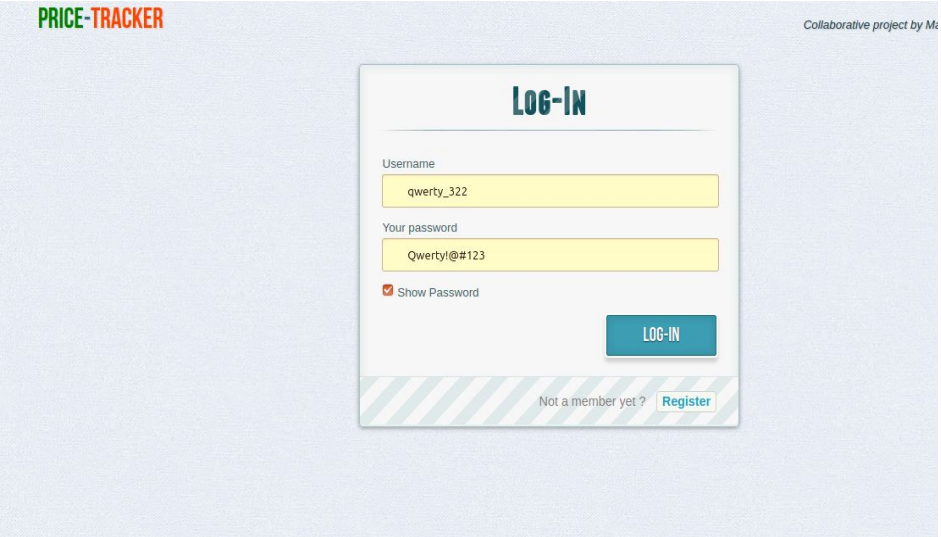
Database → MYSQL
Front End → HTML, CSS, JS, Bootstrap
Generating Charts → CHART.js, PYTHON3
Database-Front End Connectivity → PHP, jquery

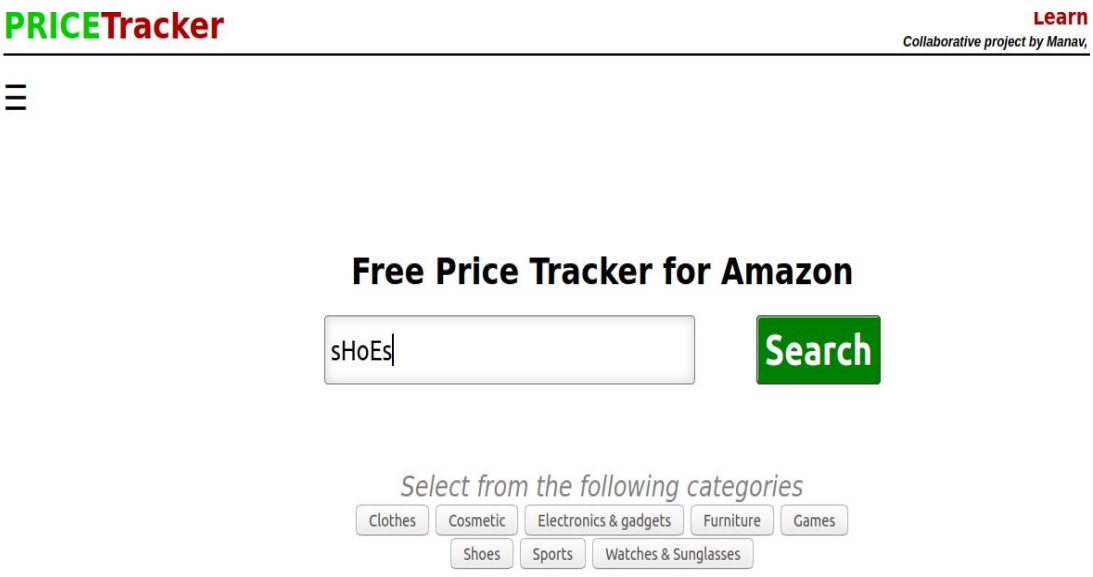
Software Used → XAMPP (WIN10 or UBUNTU 18.04)

Output Images:



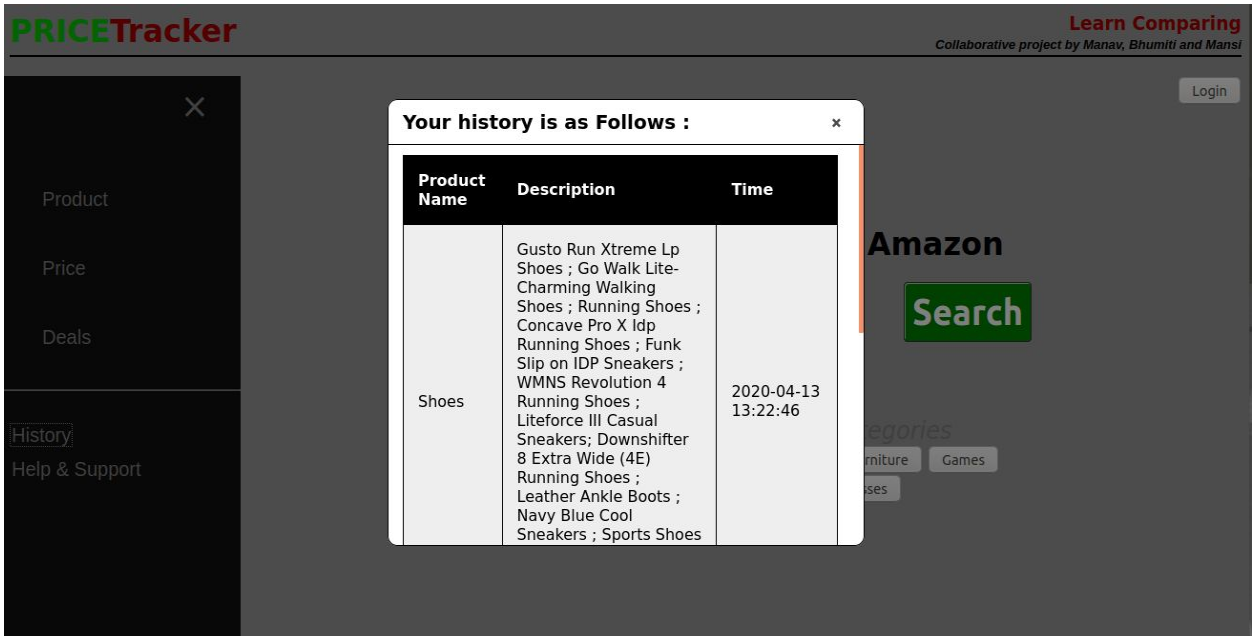
	user_id	mail_id	pswd
Edit Copy Delete	aggregorX	manavvagrecha1321@gmail.com	Qwerty!@#123
Edit Copy Delete	qazxsw_123	qazxsw@gmail.com	Qazxsw!@12
Edit Copy Delete	qwerty_322	lkjhgfdsa123@mnbvc.xz	Qwerty!@#123





Main frame : Searching for Shoes

No.	Product Name	MRP
12	Gusto Run Xtreme Lp Shoes	3600
13	Go Walk Lite-Charming Walking Shoes	3500
14	Running Shoes	2000
15	Concave Pro X Idp Running Shoes	1000
16	Funk Slip on IDP Sneakers	800
17	WMNS Revolution 4 Running Shoes	2000
18	Liteforce III Casual Sneakers	1999
19	Downshifter 8 Extra Wide (4E) Running Shoe	3000
22	Leather Ankle Boots	300
49	Navy Blue Cool Sneakers	1000
50	Sports Shoes	750
51	Sneakers shoes	460



PRICETracker

Learn Comparing

Collaborative project by Manav, Bhumiti and Mansi

☰ Sorting Filtering xxxUSERxxx LOG-IN

No.	Product Name	M. R. P.
1	Rockerz 450 Wireless Bluetooth Headphone (Luscious Black)	1500
2	iPhone 11 Pro (256GB) - Space Grey	80000
3	iPhone XR (64GB) - Black	48500
4	ZenBook Duo UX481 Intel Core i7 10th Gen 14-inch FHD Thin & Light Laptop	112000
5	Airdopes 311v2 True Wireless Earbuds with HD Sound and Charging Case (Active Black)	2999
6	BassHeads 225 in-Ear Super Extra Bass Headphones (Molten Orange)	600
7	Inspiron Core i5 8th Gen 8250U 2018	54400
8	Flip 3 Stealth Waterproof Portable Bluetooth Speaker with Rich Deep Bass (Black), Without Mic	4299

Select one of the Filter method ×

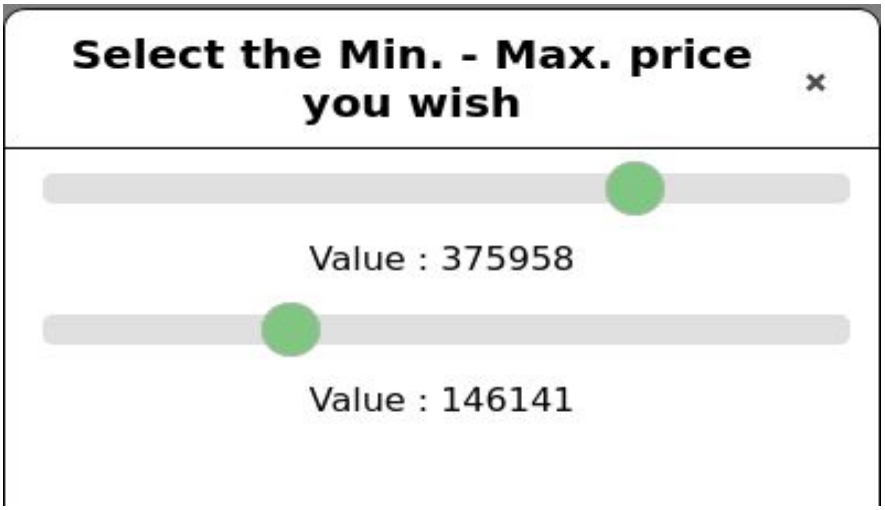
Select Brand

Select Price Range

Select Products with Deals

Select Sale


Select one of the Sort method
Sort by Prices
Sort by Name
Sort by Discount
Sort by Rating
Sort by Sell


Select the Min. - Max. price you wish
Value : 375958
Value : 146141

Add Product

Home | Update Values | Delete Product

Select Category : Games

Enter New Brand

New Brand : Santa Monika Studio

Select Brand : adidas

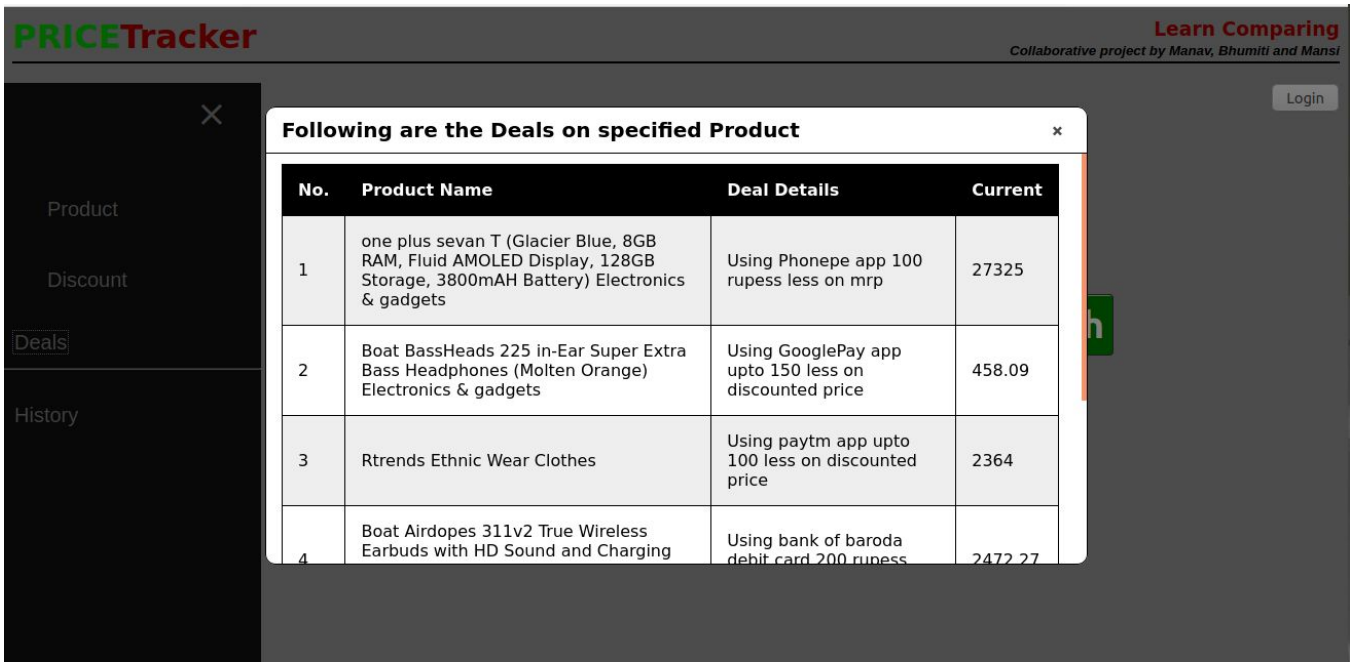
Product Name : God of War 4

Upload Image : Browse... gfw.jpeg

M. R. P. : 2990

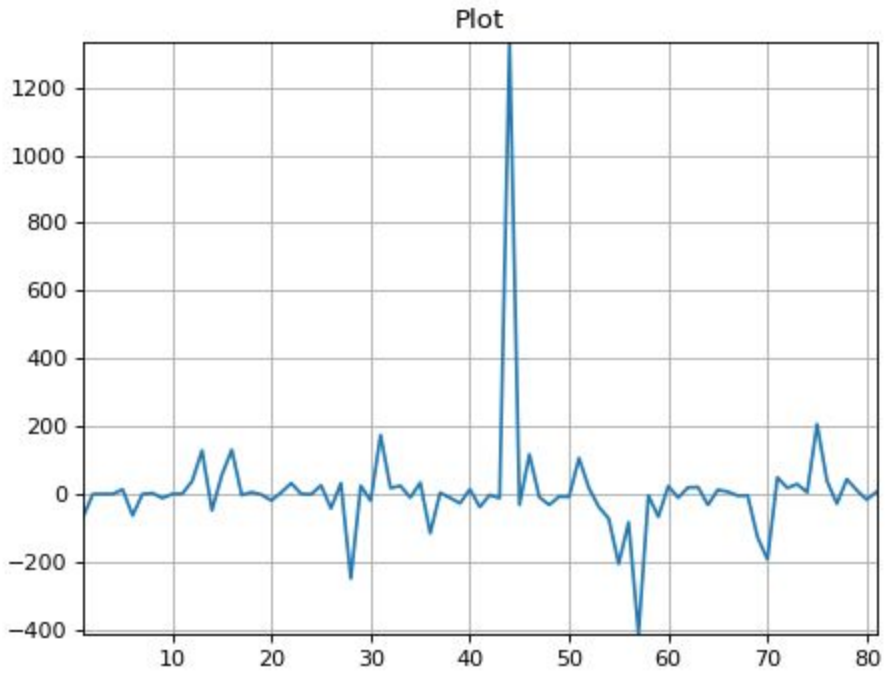
Current Price : 2499

Add product



ZenBook Duo UX481 Intel Core i7 10th Gen 14-inch FHD Thin & Light Laptop

BRAND :	ASUS
CATEGORY :	Electronics & gadgets
M. R. P. (Company Price) :	112000
Current (Discount) Price :	



This is a plot of Elasticity of Demand of Products
[We have used Stationary Data so actual value for products may differ]