

## Outputs:

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the following code:

```
INSERT ALL
  INTO users VALUES (1, 'john_doe', 'john@example.com', TO_DATE('15-JAN-2022', 'DD-MON-YYYY'), 'USA')
  INTO users VALUES (2, 'jane_smith', 'jane@example.com', TO_DATE('20-FEB-2022', 'DD-MON-YYYY'), 'UK')
  INTO users VALUES (3, 'mike_johnson', 'mike@example.com', TO_DATE('10-MAR-2022', 'DD-MON-YYYY'), 'Canada')
  INTO users VALUES (4, 'sarah_williams', 'sarah@example.com', TO_DATE('05-APR-2022', 'DD-MON-YYYY'), 'Australia')
  INTO users VALUES (5, 'david_brown', 'david@example.com', TO_DATE('12-MAY-2022', 'DD-MON-YYYY'), 'USA')
SELECT * FROM users;
```

The Results tab displays the following table:

| USERID | USERNAME       | EMAIL             | REGISTRATION_DATE | COUNTRY   |
|--------|----------------|-------------------|-------------------|-----------|
| 1      | john_doe       | john@example.com  | 15-JAN-22         | USA       |
| 2      | jane_smith     | jane@example.com  | 20-FEB-22         | UK        |
| 3      | mike_johnson   | mike@example.com  | 10-MAR-22         | Canada    |
| 4      | sarah_williams | sarah@example.com | 05-APR-22         | Australia |
| 5      | david_brown    | david@example.com | 12-MAY-22         | USA       |

5 rows returned in 0.00 seconds. CSV Export

Create users table:

The screenshot shows the Oracle Database Express Edition interface. The SQL Commands window contains the following code:

```
INSERT ALL
  INTO products VALUES (102, 'Wireless Headphones', 'Electronics', 149.99, 100)
  INTO products VALUES (103, 'Coffee Maker', 'Home Appliances', 89.99, 30)
  INTO products VALUES (104, 'Running Shoes', 'Sports', 129.99, 75)
  INTO products VALUES (105, 'Yoga Mat', 'Sports', 39.99, 120)
  INTO products VALUES (106, 'Blender', 'Home Appliances', 59.99, 40)
  INTO products VALUES (107, 'Smart Watch', 'Electronics', 199.99, 60)
SELECT * FROM products;
```

The Results tab displays the following table:

| PRODUCTID | PRODUCT_NAME        | CATEGORY        | BASE_PRICE | STOCK_QUANTITY |
|-----------|---------------------|-----------------|------------|----------------|
| 101       | Smartphone X        | Electronics     | 799.99     | 50             |
| 102       | Wireless Headphones | Electronics     | 149.99     | 100            |
| 103       | Coffee Maker        | Home Appliances | 89.99      | 30             |
| 104       | Running Shoes       | Sports          | 129.99     | 75             |
| 105       | Yoga Mat            | Sports          | 39.99      | 120            |
| 106       | Blender             | Home Appliances | 59.99      | 40             |
| 107       | Smart Watch         | Electronics     | 199.99     | 60             |

7 rows returned in 0.00 seconds. CSV Export

Create products table:

The screenshot shows the SQL Developer interface with a red title bar. The 'SQL Commands' window contains the following SQL script:

```
INTO purchases VALUES (1005, 4, 103, 'Sports', 0.10, 37.99, 'Debit Card', TO_TIMESTAMP('2023-02-11 10:28', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1006, 5, 106, 'Home Appliances', 0.10, 53.99, 'Credit Card', TO_TIMESTAMP('2023-02-15 13:45:19', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1007, 2, 107, 'Electronics', 0.25, 149.99, 'PayPal', TO_TIMESTAMP('2023-03-01 15:30:42', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1008, 3, 101, 'Electronics', 0.00, 799.99, 'Debit Card', TO_TIMESTAMP('2023-03-10 10:20:15', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1009, 1, 102, 'Electronics', 0.20, 119.99, 'Credit Card', TO_TIMESTAMP('2023-03-15 14:10:37', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1010, 4, 103, 'Home Appliances', 0.15, 76.49, 'PayPal', TO_TIMESTAMP('2023-03-20 09:30:55', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1011, 5, 104, 'Sports', 0.10, 116.99, 'Debit Card', TO_TIMESTAMP('2023-04-05 12:40:21', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1012, 2, 105, 'Sports', 0.00, 39.99, 'Credit Card', TO_TIMESTAMP('2023-04-10 16:25:48', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1013, 3, 106, 'Home Appliances', 0.05, 56.99, 'PayPal', TO_TIMESTAMP('2023-04-15 11:15:33', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1014, 1, 107, 'Electronics', 0.30, 139.99, 'Credit Card', TO_TIMESTAMP('2023-05-01 10:50:22', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1015, 4, 101, 'Electronics', 0.10, 719.99, 'Debit Card', TO_TIMESTAMP('2023-05-10 14:35:17', 'YYYY-MM-DD HH24:MI:SS'))

SELECT * FROM purchases;
```

The 'Results' window shows the following table:

| PURCHASE_ID | USERID | PRODUCTID | CATEGORY        | DISCOUNT | FINAL_PRICE | PAYMENT_METHOD | PURCHASE_DATE                |
|-------------|--------|-----------|-----------------|----------|-------------|----------------|------------------------------|
| 1001        | 1      | 101       | Electronics     | .1       | 719.99      | Credit Card    | 05-JAN-23 10 15 22.000000 AM |
| 1002        | 2      | 102       | Electronics     | .15      | 127.49      | PayPal         | 10-JAN-23 02 30 45.000000 PM |
| 1003        | 3      | 103       | Home Appliances | 0        | 89.99       | Credit Card    | 15-JAN-23 09 45 12.000000 AM |
| 1004        | 1      | 104       | Sports          | .2       | 103.99      | Debit Card     | 02-FEB-23 04 20 33.000000 PM |
| 1005        | 4      | 105       | Sports          | .05      | 37.99       | PayPal         | 10-FEB-23 11 10 28.000000 AM |
| 1006        | 5      | 106       | Home Appliances | .1       | 53.99       | Credit Card    | 15-FEB-23 01 45 19.000000 PM |
| 1007        | 2      | 107       | Electronics     | .25      | 149.99      | PayPal         | 01-MAR-23 03 30 42.000000 PM |
| 1008        | 3      | 101       | Electronics     | 0        | 799.99      | Debit Card     | 10-MAR-23 10 20 15.000000 AM |
| 1009        | 1      | 102       | Electronics     | .2       | 119.99      | Credit Card    | 15-MAR-23 02 10 37.000000 PM |
| 1010        | 4      | 103       | Home Appliances | .15      | 76.49       | PayPal         | 20-MAR-23 09 30 55.000000 AM |

More than 10 rows available. Increase rows selector to view more rows.  
10 rows returned in 0.01 seconds CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Create purchases table:

The screenshot shows the SQL Developer interface with a red title bar. The 'SQL Commands' window contains the following SQL script:

```
INTO purchases VALUES (1011, 5, 104, 'Sports', 0.10, 116.99, 'Debit Card', TO_TIMESTAMP('2023-04-05 12:40:21', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1012, 2, 105, 'Sports', 0.00, 39.99, 'Credit Card', TO_TIMESTAMP('2023-04-10 16:25:48', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1013, 3, 106, 'Home Appliances', 0.05, 56.99, 'PayPal', TO_TIMESTAMP('2023-04-15 11:15:33', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1014, 1, 107, 'Electronics', 0.30, 139.99, 'Credit Card', TO_TIMESTAMP('2023-05-01 10:50:22', 'YYYY-MM-DD HH24:MI:SS'))
INTO purchases VALUES (1015, 4, 101, 'Electronics', 0.10, 719.99, 'Debit Card', TO_TIMESTAMP('2023-05-10 14:35:17', 'YYYY-MM-DD HH24:MI:SS'))

SELECT * FROM purchases;
```

The 'Results' window shows the following table:

| PURCHASE_ID | USERNAME       | PRODUCT_NAME        | FINAL_PRICE | PURCHASE_DATE        |
|-------------|----------------|---------------------|-------------|----------------------|
| 1008        | mike_johnson   | Smartphone X        | 799.99      | 10-MAR-2023 10:20:15 |
| 1015        | sarah_williams | Smartphone X        | 719.99      | 10-MAY-2023 14:35:17 |
| 1001        | john_doe       | Smartphone X        | 719.99      | 05-JAN-2023 10:15:22 |
| 1007        | jane_smith     | Smart Watch         | 149.99      | 01-MAR-2023 15:30:42 |
| 1014        | john_doe       | Smart Watch         | 139.99      | 01-MAY-2023 10:50:22 |
| 1002        | jane_smith     | Wireless Headphones | 127.49      | 10-JAN-2023 14:30:45 |
| 1009        | john_doe       | Wireless Headphones | 119.99      | 15-MAR-2023 14:10:37 |
| 1011        | david_brown    | Running Shoes       | 116.99      | 05-APR-2023 12:40:21 |
| 1004        | john_doe       | Running Shoes       | 103.99      | 02-FEB-2023 16:20:33 |

9 rows returned in 0.01 seconds CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Get all purchases with final price > \$100:

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/f?p=4500:1003:72832788258990::NO::

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

Save Run

```

WHERE p.purchase_date BETWEEN TO_TIMESTAMP('2023-01-01', 'YYYY-MM-DD')
AND TO_TIMESTAMP('2023-03-31 23:59:59', 'YYYY-MM-DD HH24:MI:SS')
ORDER BY p.final_price DESC;

SELECT p.purchase_id, u.username, pr.product_name, p.final_price,
       TO_CHAR(p.purchase_date, 'DD-MON-YYYY') AS purchase_date
FROM purchases p
JOIN users u ON p.userid = u.userid
JOIN products pr ON p.productid = pr.productid
WHERE p.category = 'Electronics'
AND p.purchase_date BETWEEN TO_TIMESTAMP('2023-01-01', 'YYYY-MM-DD')
AND TO_TIMESTAMP('2023-03-31 23:59:59', 'YYYY-MM-DD HH24:MI:SS')
ORDER BY p.purchase_date;

```

Results Explain Describe Saved SQL History

| PURCHASE_ID | USERNAME     | PRODUCT_NAME        | FINAL_PRICE | PURCHASE_DATE |
|-------------|--------------|---------------------|-------------|---------------|
| 1001        | john_doe     | Smartphone X        | 719.99      | 05-JAN-2023   |
| 1002        | jane_smith   | Wireless Headphones | 127.49      | 10-JAN-2023   |
| 1007        | jane_smith   | Smart Watch         | 149.99      | 01-MAR-2023   |
| 1008        | mike_johnson | Smartphone X        | 799.99      | 10-MAR-2023   |
| 1009        | john_doe     | Wireless Headphones | 119.99      | 15-MAR-2023   |

5 rows returned in 0.00 seconds CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

4:06 pm 15/05/2025

Find purchases in Electronics category made in Q1 2023:

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/f?p=4500:1003:72832788258990::NO::

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

Save Run

```

AND TO_TIMESTAMP('2023-03-31 23:59:59', 'YYYY-MM-DD HH24:MI:SS')
ORDER BY p.purchase_date;

SELECT
category,
COUNT(*) AS purchase_count,
ROUND(AVG(final_price), 2) AS avg_final_price,
ROUND(AVG(discount), 2) AS avg_discount,
SUM(final_price) AS total_revenue
FROM purchases
GROUP BY category
ORDER BY total_revenue DESC;

```

Results Explain Describe Saved SQL History

| CATEGORY        | PURCHASE_COUNT | AVG_FINAL_PRICE | AVG_DISCOUNT | TOTAL_REVENUE |
|-----------------|----------------|-----------------|--------------|---------------|
| Electronics     | 7              | 396.78          | .16          | 2777.43       |
| Sports          | 4              | 74.74           | .09          | 298.96        |
| Home Appliances | 4              | 69.37           | .08          | 277.46        |

3 rows returned in 0.00 seconds CSV Export

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

4:06 pm 15/05/2025

Average final price and discount by category:

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/?p=4500:1003:72832788258990:NO...

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
ORDER BY total_revenue DESC;
```

```
SELECT
  EXTRACT(YEAR FROM purchase_date) AS year,
  EXTRACT(MONTH FROM purchase_date) AS month,
  COUNT(*) AS total_orders,
  COUNT(DISTINCT userid) AS unique_customers,
  SUM(final_price) AS total_revenue,
  ROUND(AVG(final_price), 2) AS avg_order_value
FROM purchases
GROUP BY EXTRACT(YEAR FROM purchase_date), EXTRACT(MONTH FROM purchase_date)
ORDER BY year, month;
```

Results Explain Describe Saved SQL History

| YEAR | MONTH | TOTAL_ORDERS | UNIQUE_CUSTOMERS | TOTAL_REVENUE | AVG_ORDER_VALUE |
|------|-------|--------------|------------------|---------------|-----------------|
| 2023 | 1     | 3            | 3                | 937.47        | 312.49          |
| 2023 | 2     | 3            | 3                | 195.97        | 65.32           |
| 2023 | 3     | 4            | 4                | 1146.46       | 286.62          |
| 2023 | 4     | 3            | 3                | 213.97        | 71.32           |
| 2023 | 5     | 2            | 2                | 859.98        | 429.99          |

5 rows returned in 0.02 seconds CSV Export

Application Express 21.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Monthly sales performance:

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/?p=4500:1003:72832788258990:NO...

Sign in

```
SELECT
  u.username,
  u.email,
  TO_CHAR(p.purchase_date, 'DD-MON-YYYY') AS purchase_date,
  pr.product_name,
  pr.category,
  p.final_price,
  p.payment_method
FROM purchases p
JOIN users u ON p.userid = u.userid
JOIN products pr ON p.productid = pr.productid
ORDER BY u.username, p.purchase_date DESC;
```

Results Explain Describe Saved SQL History

| USERNAME     | EMAIL             | PURCHASE_DATE | PRODUCT_NAME        | CATEGORY        | FINAL_PRICE | PAYMENT_METHOD |
|--------------|-------------------|---------------|---------------------|-----------------|-------------|----------------|
| david_brown  | david@example.com | 05-APR-2023   | Running Shoes       | Sports          | 116.99      | Debit Card     |
| david_brown  | david@example.com | 15-FEB-2023   | Blender             | Home Appliances | 53.99       | Credit Card    |
| jane_smith   | jane@example.com  | 10-APR-2023   | Yoga Mat            | Sports          | 39.99       | Credit Card    |
| jane_smith   | jane@example.com  | 01-MAR-2023   | Smart Watch         | Electronics     | 149.99      | PayPal         |
| jane_smith   | jane@example.com  | 10-JAN-2023   | Wireless Headphones | Electronics     | 127.49      | PayPal         |
| john_doe     | john@example.com  | 01-MAY-2023   | Smart Watch         | Electronics     | 139.99      | Credit Card    |
| john_doe     | john@example.com  | 15-MAR-2023   | Wireless Headphones | Electronics     | 119.99      | Credit Card    |
| john_doe     | john@example.com  | 02-FEB-2023   | Running Shoes       | Sports          | 103.99      | Debit Card     |
| john_doe     | john@example.com  | 05-JAN-2023   | Smartphone X        | Electronics     | 719.99      | Credit Card    |
| mike_johnson | mike@example.com  | 15-APR-2023   | Blender             | Home Appliances | 56.99       | PayPal         |

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds CSV Export

Application Express 21.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Customer purchase history with product details:

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/f?p=4500:1003:72832788258990::NO::

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
JOIN products pr ON pr.productid = p.productid
ORDER BY u.username, p.purchase_date DESC;

SELECT
  pr.productid,
  pr.product_name,
  pr.category,
  pr.base_price
FROM products pr
LEFT JOIN purchases p ON pr.productid = p.productid
WHERE p.productid IS NULL;
```

Results Explain Describe Saved SQL History

no data found

Language: en-us

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

4:09 pm  
15/05/2025

*Products never purchased (using LEFT JOIN):*

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/f?p=4500:1003:72832788258990::NO::

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10 Save Run

```
SELECT
  pr.productid,
  pr.product_name,
  ROUND(AVG(p.discount), 2) AS avg_discount,
  COUNT(p.purchase_id) AS times_purchased
FROM products pr
JOIN purchases p ON pr.productid = p.productid
GROUP BY pr.productid, pr.product_name
HAVING AVG(p.discount) > (SELECT AVG(discount) FROM purchases)
ORDER BY avg_discount DESC;
```

Results Explain Describe Saved SQL History

| PRODUCTID | PRODUCT_NAME        | AVG_DISCOUNT | TIMES_PURCHASED |
|-----------|---------------------|--------------|-----------------|
| 107       | Smart Watch         | 28           | 2               |
| 102       | Wireless Headphones | 18           | 2               |
| 104       | Running Shoes       | 15           | 2               |

3 rows returned in 0.00 seconds [CSV Export](#)

Language: en-us

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

4:09 pm  
15/05/2025

*Products with above-average discounts:*

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/f?p=4500:1003:72832788258990::NO::

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

Save Run

```

SELECT
  u.username,
  p.productid,
  pr.product_name,
  p.final_price,
  TO_CHAR(p.purchase_date, 'DD-MON-YYYY HH24:MI:SS') AS purchase_date
FROM purchases p
JOIN users u ON p.userid = u.userid
JOIN products pr ON p.productid = pr.productid
WHERE (p.userid, p.purchase_date) IN (
  SELECT userid, MAX(purchase_date) IN (
    FROM purchases
    GROUP BY userid
  )
)
ORDER BY u.username;

```

Results Explain Describe Saved SQL History

| USERNAME       | PRODUCTID | PRODUCT_NAME  | FINAL_PRICE | PURCHASE_DATE        |
|----------------|-----------|---------------|-------------|----------------------|
| david_brown    | 104       | Running Shoes | 116.99      | 05-APR-2023 12:40:21 |
| jane_smith     | 105       | Yoga Mat      | 39.99       | 10-APR-2023 16:25:48 |
| john_doe       | 107       | Smart Watch   | 139.99      | 01-MAY-2023 10:50:22 |
| mike_johnson   | 106       | Blender       | 56.99       | 15-APR-2023 11:15:33 |
| sarah_williams | 101       | Smartphone X  | 719.99      | 10-MAY-2023 14:35:17 |

5 rows returned in 0.01 seconds CSV Export

Application Express 2.1.0.0.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Language: en-us

4:10 pm  
15/05/2025

Most recent purchase for each customer:

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/f?p=4500:1003:72832788258990::NO::

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

Autocommit Display 10

Save Run

```

SELECT
  TO_CHAR(cohort_month, 'YYYY-MM') AS cohort,
  TO_CHAR(activity_month, 'YYYY-MM') AS month,
  month_number,
  active_users,
  FIRST_VALUE(active_users) OVER (PARTITION BY cohort_month ORDER BY activity_month) AS cohort_size,
  ROUND(active_users / FIRST_VALUE(active_users) OVER (PARTITION BY cohort_month ORDER BY activity_month) * 100, 1) AS retention_rate
FROM monthly_activity
ORDER BY cohort_month, activity_month;

```

Results Explain Describe Saved SQL History

| COHORT  | MONTH   | MONTH_NUMBER | ACTIVE_USERS | COHORT_SIZE | RETENTION_RATE |
|---------|---------|--------------|--------------|-------------|----------------|
| 2023-01 | 2023-01 | 0            | 3            | 3           | 100            |
| 2023-01 | 2023-02 | 1            | 1            | 3           | 33.3           |
| 2023-01 | 2023-03 | 2            | 3            | 3           | 100            |
| 2023-01 | 2023-04 | 3            | 2            | 3           | 66.7           |
| 2023-01 | 2023-05 | 4            | 1            | 3           | 33.3           |
| 2023-02 | 2023-02 | 0            | 2            | 2           | 100            |
| 2023-02 | 2023-03 | 1            | 1            | 2           | 50             |
| 2023-02 | 2023-04 | 2            | 1            | 2           | 50             |
| 2023-02 | 2023-05 | 3            | 1            | 2           | 50             |

9 rows returned in 0.01 seconds CSV Export

4:14 pm  
15/05/2025

Customer cohort retention analysis:

SQL Commands

TASK 3 DATA ANALYST[1].pdf

127.0.0.1:8080/apex/?p=4500:1003:72832788258990:NO...

Autocommit Display 10 Save Run

```
ROUND(AVG(final_price), 2) AS avg_order_value,  
COUNT(DISTINCT userid) AS unique_customers  
FROM purchases  
GROUP BY category,  
CASE  
WHEN discount = 0 THEN 'No discount'  
WHEN discount < 0.1 THEN '0-10%'  
WHEN discount < 0.2 THEN '10-20%'  
WHEN discount < 0.3 THEN '20-30%'  
ELSE '30%+'  
END  
ORDER BY category, discount_range;
```

Results Explain Describe Saved SQL History

| CATEGORY        | DISCOUNT_RANGE | TRANSACTION_COUNT | TOTAL_REVENUE | AVG_ORDER_VALUE | UNIQUE_CUSTOMERS |
|-----------------|----------------|-------------------|---------------|-----------------|------------------|
| Electronics     | 10-20%         | 3                 | 1567.47       | 522.49          | 3                |
| Electronics     | 20-30%         | 2                 | 269.98        | 134.99          | 2                |
| Electronics     | 30%+           | 1                 | 139.99        | 139.99          | 1                |
| Electronics     | No discount    | 1                 | 799.99        | 799.99          | 1                |
| Home Appliances | 0-10%          | 1                 | 56.99         | 56.99           | 1                |
| Home Appliances | 10-20%         | 2                 | 130.48        | 65.24           | 2                |
| Home Appliances | No discount    | 1                 | 89.99         | 89.99           | 1                |
| Sports          | 0-10%          | 1                 | 37.99         | 37.99           | 1                |
| Sports          | 10-20%         | 1                 | 116.99        | 116.99          | 1                |
| Sports          | 20-30%         | 1                 | 103.99        | 103.99          | 1                |

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.00 seconds [CSV Export](#)

Application Express 2.1.0.00.39  
Copyright © 1999, 2006, Oracle. All rights reserved.

Language: en-us

4:15 pm  
15/05/2025

*Discount effectiveness by product category:*